DM7,- Sfr7,- ÖS56,- März '88 Nr.3

GAMIGA

KICK START

BESUCHEN SIE UNS
HANNOVER MESSE
CeBIT '88
Welt-Centrum Büro · Information · Telekommunikation
16. – 23. MÄRZ 1988
HALLE 17 / STAND A 70



DER GRAFIK JONGLEUR

SUPERGRAFIK SUBSYSTEM FÜR AMIGA

C-64 Emulator · Mit Basic an's System · Boot Block
PC-Intim · Tips und Tricks · Top 12 · Lattice · C 4.0

EDITORIAL



DEM VER-
DANKT IHR
DAS
LAYOUT
DIESES
BLATTES.
(UND DES
TITELS)

DIE ÄRGERT
SICH,
DASS ES BUNTE
KLAMOTTEN
NICHT
SCHWARZ/WEISS
ZU
KAUFEN GIBT.

DER FAND
DAS ALLES
ZIEMLICH
LANGWEILIG.

DER KANN
ÜBERHAUPT
NICHT
JONGLIEREN.

DER MACHT
GANZ EINFACH
GUTE FOTOS.

VIEL VERGNÜGEN,

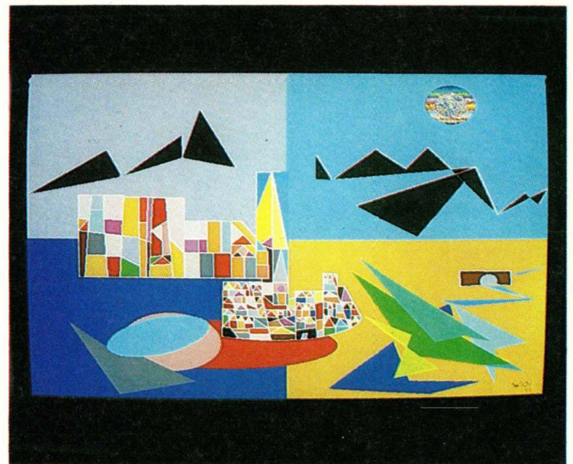
M. Mayer

INHALT

DER GRAFISCHE GEHILFE

Die Auflösung des AMIGA galt schon immer als grandios. Doch das Grafik-Subsystem MEGAVISION 2 zeigt, daß dies noch lange nicht genug sein muß. Mit einem eigenen Monitor versehen, verwandelt sie den AMIGA in ein semiprofessionelles Grafiksystem, das einiges zu bieten hat.

Seite 14



KICKS FÜR INSIDER

In der zweiten KICKS-Ausgabe sind auch wieder interessante KICKS enthalten, so z.B. der Röhrenschoner, der den Bildschirm nach einer bestimmten Zeit erlöschen läßt.

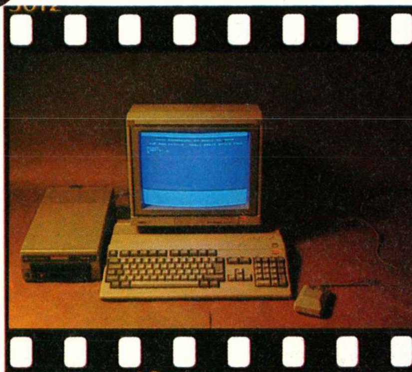
Mehr auf Seite 83



3/88

64ER-FIEBER

Nachdem der AMIGA nun fast jeden angesteckt hat, ensteht die Frage, ob die User gegen das neuentdeckte 64er-Fieber immun sind.
Wir waren jedenfalls dagegen gerüstet. Seite **80**



MIT BASIC ANS SYSTEM

Die Sprache Basic wird von ernsthaften Programmierern aus vielerlei Gründen gerne belächelt. Ein Argument - der unklare Systemzugriff - möchten wir mit dieser Serie entkräften. Ran geht's. Seite **49**



PC-INTIM

Im ersten Teil unserer neuen Serie gehen wir auf die Grafik des PC ein. Alle Sidecar- bzw. Bridgeboard-User kommen hier auf ihre Kosten. Seite **21**

KICKS FÜR INSIDER

SPIELE

RUBRIK

VERSION 3.6
AZTEC-C68 FÜR AMIGA

FABIAN & MAYER

Wußten Sie, daß eines der verbreitesten und komplexesten Betriebssysteme - UNIX - in C geschrieben ist ?

Wußten Sie, daß auch das Betriebssystem des AMIGA größtenteils in C geschrieben wurde ?

Wußten Sie, daß C eine der wichtigsten und modernsten Programmiersprachen ist ?

Wußten Sie, daß Aztec-C einer der schnellsten und leistungsfähigsten Compiler für den AMIGA ist ?

Wußten Sie , daß jetzt Aztec-C in der Version 3.6 verfügbar ist ?

Möchten Sie mehr darüber wissen? Dann schicken Sie uns einen ausreichend frankierten Rückumschlag und Sie erhalten ausführliche Information.

HIERMIT BESTELLE ICH:

- ☐ AZTEC-C68K/AM-P PROFESSIONAL SYSTEM FÜR DM398.-
- ☐ AZTEC-C68K/AM-D DEVELOPER SYSTEM FÜR DM598.-
- ☐ AZTEC-SDB SOURCE LEVEL DEBUGGER FÜRDM149.-

VERSANDKOSTEN:
INLAND DM 7,50, AUSLAND DM 10,-
VORAUSKASSE
AUSLANDSBESTELLUNGEN NUR GEGEN VORAUSKASSE

☐ GEGEN VORAUSKASSE
☐ NACHNAHME
(3,70 DM NACHNAHMEGEBÜHR)

NAME: ____________________

VORNAME: ____________________

STRAßE: ____________________

ORT: ____________________

UNTERSCHRIFT: ____________________



MERLIN COMPUTER GMBH
INDUSTRIESTRAßE 26
6236 ESCHBORN
TEL. 06196/481811

NEUE DRUCKER VON STAR

Der Drucker-Riese Star kommt wieder ins Gerede. Das Low-Cost-Produkt aus diesem Hause, der NL-10, wird durch einen neuen Einstiegsdrucker ergänzt. Laut Star-Management soll der neue Drucker LC-10 an die Verkaufserfolge des NL-10 anknüpfen. Der Preis liegt mit 695.- DM in denselben Breiten wie der seines Vorgängers. Natürlich präsentiert sich der LC-10 mit einigen Neuerungen. Diesbezüglich ist die dem Trend folgende 'Papier-Park-Einrichtung' zu nennen, das bedeutet, daß bei einem Wechsel von Endlos- auf Einzelblattpapier das Endlospapier nicht aus dem Drucker entfernt werden muß, die einzelnen Blätter werden über einen separaten Schacht eingeführt. Eine weitere Neuerung gegenüber dem Vorgänger besteht im Papiervorschub, der jetzt auch in kleineren Schritten vollzogen werden kann. Außerdem sind vier standardmäßig eingebaute Fonts erreichbar - Courier, Sanserif, Orator 1 + 2. Auf der Basis eines 9 Nadel-Matrix-Druckers besitzt er folgende technische Daten: 144 cps bei Schnell-, 36 cps bei Schönschrift, 10 Zoll Druckbreite, vier verschiedene Fonts, paralleles Centronics-Interface (serielles Commodore-Interface für C64), Schubtraktor, Papier-Park-Einrichtung, halbautomatischer Einzelblatteinzug, 4 KB Speicher und eine Geräuschentwicklung von 53 dB(A).Für 245.- DM ist ein automatischer Einzelblatteinzug als Zubehör erhältlich.



Der neue 9-Nadel-Matrix-Drucker von Star, der LC-10

Für die Zukunft wird auch eine Farbversion des LC-10 verfügbar sein - der LC-10 Color. Äußerlich gleicht er dem LC-10 und wird bis zu sieben Farben auf das Papier bringen. Zum Preis von 795.- DM wird das Farb-Gerät ab März '88 im Fachhandel erhältlich sein.

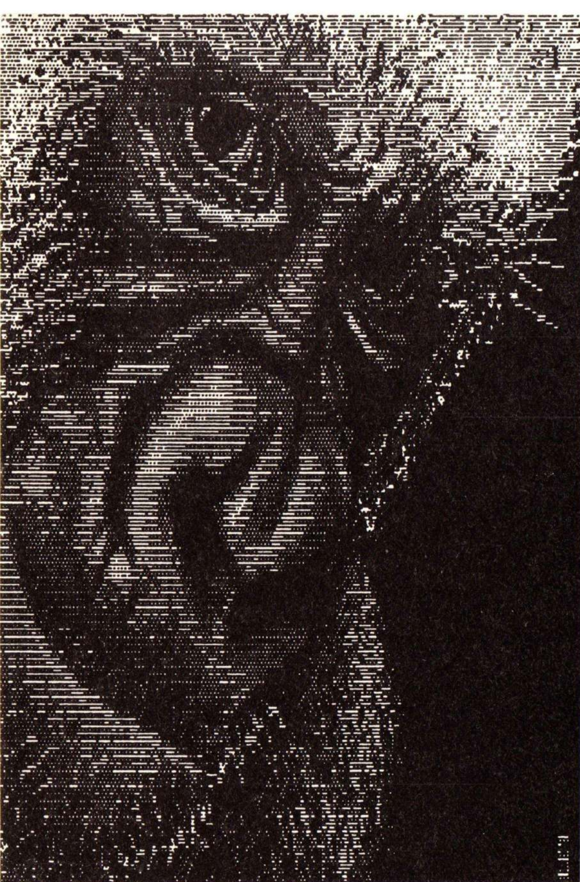
Zur LC-Familie gesellt sich auch ein 24 Nadler - der LC24-10. Gegenüber dem 9 Nadel-Drucker bietet der LC24-10 eine höhere Druckgeschwindigkeit und ein besseres Schriftbild. Neben der Papier-Park-Einrichtung ist eine weitere Neuerung eingebaut, eine sogenannte 'Leise-Taste'. Bei Bedarf kann der Geräuschpegel des Druckers gesenkt werden.

STAR

MICONISCS DEUTSCHLAND GmbH

Mergenthalerallee 1-3

6236 Eschborn



Der Gorilla als Hardcopy, mit dem CP-80X erstellt

DER CP-80 X DRUCKER AM AMIGA

Einer der meistverkauften Drucker für den C64 ist mit Sicherheit der CP-80X. Leider gab es Schwierigkeiten, diesen Drucker an den AMIGA anzuschließen, so daß besonders Umsteiger, die diesen Drucker besitzen, verärgert waren. Die Firma CSJ COMPUTERSOFT JONIGK bietet unter anderem ein Drucker-Anpassungsset an, um eben diesen Drucker an den AMIGA anzuschließen. Im Lieferumfang befindet sich eine Workbench-Diskette, auf der sich zwei CP-80X Druckertreiber befinden, des weiteren findet der Käufer ein Kopierprogramm, mit dem die Treiber auf verschiedene Arbeitsdisketten (DPaint, Pagesetter,...) kopiert werden können.

Außerdem wird ein EPROM mitgeliefert, das gegen das im Drucker befindliche ausgetauscht werden muß. Der diesbezügliche Umbau wird in einer Bedienungsanleitung ausfürlich beschrieben, so daß auch Laien keine Schwierigkeiten zu erwarten haben. Da die Druckqualitäten des CP-80X mit dem Treiber ausgezeichnet sind und auch alle Druckmodi unterstützt werden, lohnt sich der Umbau durchaus. Der CP-80X läuft dann mit allen AMIGA.

Anbieter: CSJ COMPUTERSOFT JONIGK

An der Tiefenriede 27

3000 Hannover 1

Tel. 0511-886383

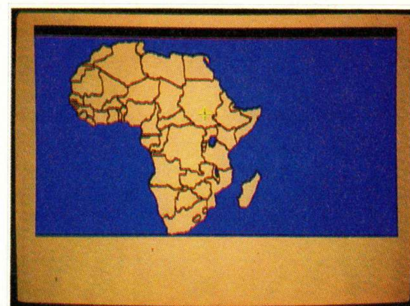
Preis: 59.90 DM

COMMODORE BIETET SOFTWARE-PAKET AN

Commodore hat zwei deutsche Programme zu einem Paket verschnürt. Es handelt sich hierbei um ein Datenbank- und ein Kalkulations-Programm. Besitzer eines AMIGA 500 oder 2000 können in den Genuß der zwei Programme mit einen Unkostenbeitrag von 399.- DM kommen. Das Kalkulationsprogramm nennt sich AmigaCalc und wird in dieser Ausgabe ausführlich getestet. Die Datenbank besitzt den Namen Superbase Personal, beide Programme haben eine mausorientierte Oberfläche und sind wie die mitgelieferten Handbücher in deutscher Sprache gehalten.

Anbieter: gut sortierte Fachhändler

Preis: 399.- DM



Eindeutige Grafik und keine überflüssigen Worte prägen das Lernprinzip.

LÄNDER DIESER ERDE

Lernen fällt meist schwer und hat nicht immer den gewünschten Erfolg. Das Programm 'Länder dieser Erde' ist ein Lernprogramm, um die vielen, vielen Länder, die es nun mal auf diesem Planeten gibt, kennenzulernen. Dazu werden die einzelnen Erdteile auf dem Bildschirm dargestellt und die Länder per Mausklick abgefragt. Die Arbeitsweise beruht auf dem Prinzip des 'gehirn-gerechten Lernens' von Vera F. Birkenbihl, die auch durch mehrere Bücher über jenes Thema bekannt wurde.

Der Preis des Produktes liegt bei DM 39.-.

Vertrieb:

LERN PARTNER

Jahnstraße 9/1

7535 Königsbach-Stein

Tel.: 07232-4293

NEUE GRAFIK SOFTWARE

THE DIRECTOR

The Director ist ein neues Animationsprogramm für den Amiga, das für die Erstellung von allen Arten von Animationen und Präsentation gedacht ist. Es stellt dem Anwender eine Art Basic-ähnliche Sprache zur Verfügung, mit der die Abläufe der Animationen einfach beschrieben werden können. Hierbei sind auch Zugriffe auf Blitterfunktionen möglich. The Director arbeitet mit sämtlichen Bildschirmmodi des Amiga und beherrscht auch den Overscan (volle Ausnutzung des Bildschirms ohne Rand links und rechts). Neben einigen standardmäßigen Bildverarbeitungsfunktionen kann der Director auch komplette IFF-Animationsfiles (z.B. mit Videoscape 3D erzeugt) einbinden, wobei die Möglichkeit besteht, diese nur in Teilbereichen des Bildschirms ablaufen zu lassen. Zusätzlich können die Animationen auch mit digitalisierten Klängen unterlegt werden.

Alles in allem verspricht dieses Programm größtmögliche Flexibilität für die mannigfaltigsten Anwendungen.

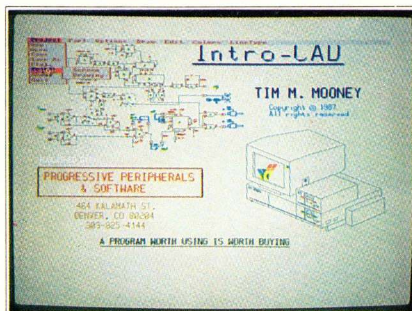
The Director

Herst.: The Right Answers Group

Vertrieb: GTI

INTROCAD

IntroCAD ist eines der neuen Produkte von Progressive Periphals & Software. Es handelt sich dabei um ein preiswertes CAD-Programm für Konstruktionszeichnungen, Schaltplanlayouts und ähnliche Anwendungen. Trotz seines niedrigen Preises bietet IntroCAD eine große



Auswahl an Funktionen, wozu unter anderem die Möglichkeit gehört, sich eine Bibliothek an Bauteilen oder anderen häufig verwendeten Zeichen zu definieren und abzuspeichern. Weiterhin sind alle notwendigen Zeichenfunktionen, beliebige Textgrößen, Rotationen und weitere spezielle CAD-Funktionen integriert. Die größte Besonderheit von IntroCAD liegt allerdings in der außerordentlich großen Auswahl an speziellen Druckertreibern, mit denen sich fast jedes gängige Modell ansteuern läßt. Diese Druckertreiber unterstützen den High Density-Modus (sofern der angesteuerte Drucker darüber verfügt), bei dem Auflösungen von bis zu 360*360 Punkte pro Zoll wiedergegeben werden können, wodurch man mit Matrixdruckern annähernd Laserdruckerqualität erreicht.

IntroCAD

Herst.: Progressive Periphals & Software

Preis: ca. 190.- DM

Vertrieb: Fachhändler

PIXMATE

PIXmate ist da zweite neue Produkt von PPS (siehe IntroCAD). Es handelt sich dabei um ein Bildverarbeitungsprogramm in der Art des Butchers. Der sogenannte Image-Processor dieses Programms soll laut Produktinformation über mehr als 3000 spezielle Effekte zur Bildverarbeitung verfügen. Weiterhin arbeitet das Programm mit speziellen neuen Blitterroutinen, die größtmögliche Geschwindigkeit mit sich bringen. Das gleiche gilt für Color Cyclings, die laut Produktinformation die schnellsten auf dem gesamten Softwaremarkt sein sollen. Auch ein Farbequalizer, mit dem man die Intensität aller Farbanteile regulieren kann, fehlt nicht. Ein Test dieses Programms wird in unserem in Kürze erhältlichen Sonderheft erscheinen.

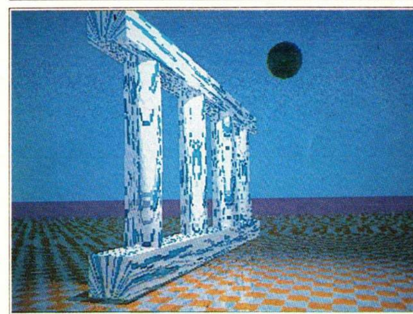
PIXmate

Herst.: Progressive Periphals & Software

Preis: ca. 170.- DM

Vertrieb: Fachhändler

SILVER 1.1



Das bereits in der Dezemberausgabe getestete Ray Tracing-Animationsprogramm Silver liegt in einer Update-Version vor, die einige neue Features enthält. So können jetzt Bilder im IFF-Format abgespeichert werden, was die Nachbearbeitung mit beliebigen Malprogrammen o.ä. ermöglicht. Die bedeutendste Neuerung ist allerdings die Möglichkeit, Objekte mit einem

beliebigen Muster zu überziehen (Texture Mapping), das als IFF-Bild oder -Brush geladen werden kann. Diese Funktion erweitert die gestalterischen Möglichkeiten von Silver enorm. Jeder Szene können bis zu 8 verschiedene Muster eingeladen und um beliebige Objekte gewickelt werden. Weiterhin wird Silver in Kürze in einer komplett deutschen Version ausgeliefert.

Silver

Herst.: Impulse, Inc.

Preis: 279.- DM (englisch), 299.- DM (deutsch)

Vertrieb: IM, Frankfurt, 069/7071102

PORTS OF CALL

Mit PORTS OF CALL präsentiert Aegis nach ARAZOKS TOMB bereits das zweite Spiel. Es handelt sich dabei um eine Schiffahrts-Handels-Simulation mit bis zu vier Spielern. Die Graphik des Spiels stammt übrigens von Sachs, der schon bei DEFENDER OF THE CROWN von Cinemaware Pate gestanden hat.



Ports of Call-Aegis läßt seine Handelsmarine auslaufen

Ziel von PORTS OF CALL ist es, eine möglichst große und ertragsreiche Handelsflotte aufzubauen. Vor den Erfolg haben die Programmierer aber zahlreiche Eisberge, Piraten und Windhosen gesetzt. Noch schlimmer wird es, wenn die Lotsen, die Ihr Schiff gegen eine entsprechende Gebühr in den Hafen steuern, beschließen zu streiken. Schäden an Kaimauern werden übrigens nicht von der Versicherung gedeckt.

Schon der Anfang des Spiels bietet genügend Optionen, um sich "ewig lange" aufzuhalten. So präjudiziert bereits die Entscheidung, welche Schiffe Sie am Anfang erwerben, zum Teil die Strategie Ihres späteren Handels und Ansehens. Die Wahl des Heimathafens (aus über 30 angebotenen Städten) kann Ihnen bereits am Anfang einen Wettbewerbsvorteil bringen (z.B. NEW YORK oder San Francisco).

Unser erster Eindruck: Gute Graphik und Spielwitz machen PORTS OF CALL zu einem sehr empfehlenswerten Spiel. Ein ausführlicher Test folgt.

Anbieter: AEGIS

THUNDERBOY



Julian auf dem Weg zum Vulkanschloß

Alles ist friedlich im Donnertal, bis sich eines Tages der Himmel verdunkelt, es anfängt zu regnen, Blitze vom Himmel zucken - kurz gesagt, es herrscht Untergangsstimmung. Drago ist aufgetaucht und unterdrückt seit diesem legendären Tag das Donnertal, nur Julian und Jane können den Einflüssen des Drachen entrinnen. Doch Drago entführt Jane in sein Vulkanschloß und Julian macht sich auf den Weg, Jane aus den Fängen des Drachen zu befreien. Alle Tiere des Tales sind dem Unhold aber untertan und machen Julian Schwierigkeiten. Soweit zur Rahmenhandlung von THUNDERBOY. Als Spieler muß man natürlich unseren Helden Julian steuern, dabei scrollt der Bildschirm von rechts nach links und Julian muß Tiere und Abgründe überspringen, außerdem sollte er sich stärken und Früchte einsammeln. Die Grafik ist zum Teil leicht naiv und kommt im Grunde nicht an höhere AMIGA-Ansprüche heran. Der Sound des Spiels ist gut, besonders wenn es unseren kleinen Helden 'erwischt' hat, ertönt ein nervenaufreibendes Gejammere, auch die Hintergrundmusik ist sauber digitalisiert und hörenswert.

THUNDERBOY wird nicht so schnell langweilig, der ständig steigende Schwierigkeitsgrad zeichnet das Spiel des weiteren aus. AMIGA-Besitzer, die solche Art von Spielen lieben, werden Gefallen an THUNDERBOY finden.

Anbieter: RAINBOW ARTS Software GmbH

Münsterstr. 27

4830 Gütersloh 1

Preis: 59.95 DM

THE WALL



The Wall hat Bilder als Hintergrundgrafiken

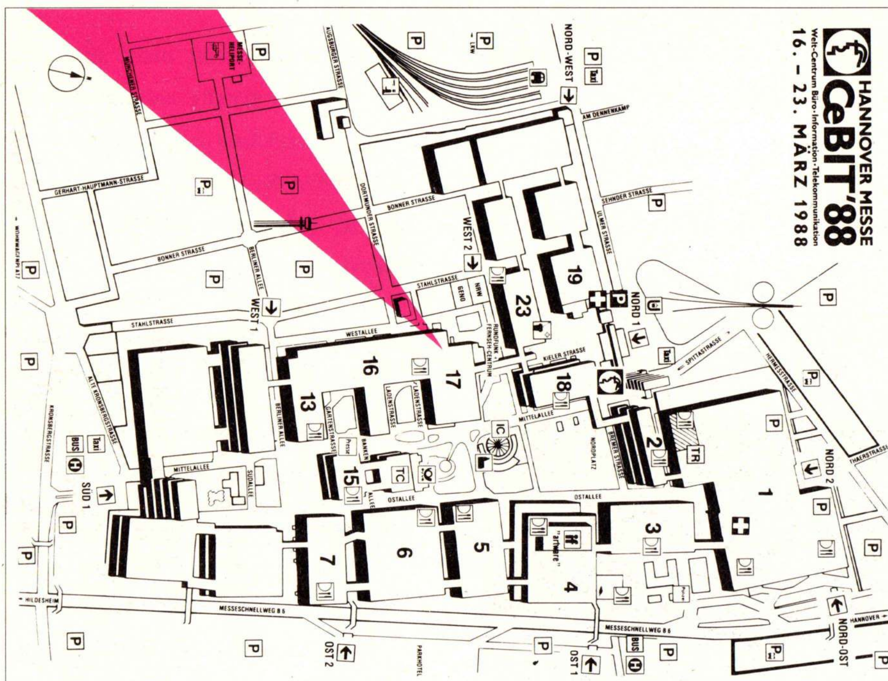
Nein, kein Bezug zu der wohl allseits bekannten LP von Pink Floyd, sondern eine weitere Breakout-Variante. Das Besondere an THE WALL ist sicherlich die gute Musik und daß Bilder als Hintergrundgrafiken dienen. Während aus dem einen Kanal Spielgeräusche erklingen, tönt aus dem zweiten eine futuristische Melodie. Des weiteren bietet THE WALL für besonders gute Spieler eine Bonusrunde an, in der ausgewählte Grafiken zum Einsatz kommen. Ansonsten bietet das Programm nichts Neues an Breakout-Variationen.

Anbieter: RAINBOW ARTS Software GmbH
Münsterstr. 27
4830 Gütersloh 1

Preis: 49.95 DM

CeBIT AKTUELL

ALLE LESER SIND HERZ-LICH WILL-KOM-MEN!



Wieder einmal ist es soweit, die größte Computermesse der Welt öffnet vom 16.-23. März ihre Hallen, um dem begeisterten Publikum das Neueste vom Computermarkt zu präsentieren. Wie immer ist der Ort der Handlung die CeBIT in Hannover, die seit Ihrer Entstehung aus der gemeinsamen Hannovermesse immer mehr an Bedeutung gewinnt.

Wie schon im letzten Jahr sind wir wieder auf dieser Messe vertreten, erstmals auch mit einem eigenen Stand, auf dem Sie mit uns Redakteuren und anderen Computerfreunden angeregte Gespräche führen können. Für Anregungen und Vorschläge haben wir immer ein offenes Ohr und auch angehende Autoren können hier erste Kontakte mit uns knüpfen. Versäumen Sie also nicht, bei Ihrem Besuch der Hannovermesse bei uns vorbeizukommen.

BESUCHEN SIE UNS! HALLE 17, STAND A70

VAMPIRE'S EMPIRE

Van Helsing, der in der Welt verachtet und gemieden ist, macht sich auf, um diese zu retten. Sein Ziel sind die dunklen Gewölbe der Finsternis, das Zuhause des Grafen DRACULA. Seine Aufgabe besteht darin, einen göttlichen Lichtstrahl in die Gemächer des Grafen zu leiten, damit der Blutsauger zu Asche zerfällt. Doch bis in die Gemächer ist es ein weiter Weg. Einige Hilfsmittel stehen dem wackeren Helden zur Verfügung - Knoblauch, Spiegel und einiges mehr.

Van Helsing ist nicht mehr der Jüngste, trotz seines dem Alters und gibt er sich aber recht sportlich, böse Geister werden mit Fußtritten verjagt, aber Detlef - ein blondgelockter Blutsauger, der sich in Van Helsing verliebt hat, läßt sich nicht so einfach abschütteln, genauso wie Sybille - die anziehende Sirene der Unterwelt.

Die Grafik von VAMPIRE'S EMPIRE ist ausgezeichnet, mit viel Liebe im Detail wurde gezeichnet und animiert. Der Sound ist der gruseligen Atmosphäre des Spieles angepaßt. Das rasante Aktionspiel wird sicher seine Fans finden. Der Redaktion hat es auf jeden fall viel Spaß bereitet.

Anbieter: ARIOLASOFT
Goethestr. 1
4830 Gütersloh
Preis: 59.95 DM

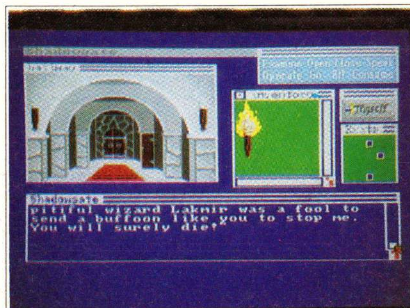


Van Helsing auf der Jagd auf Graf Dracula.

SHADOWGATE

Nach DEJA VU und UNINVITED präsentiert Mindscape jetzt mit SHADOWGATE das dritte Adventure von ICOM Simulations. Die Geschichte ist, wie bei den beiden anderen Abenteuern dieser Serie, dem Genre der FANTASY entlehnt. Drachen, Dämonen, Zauberer, Troll und Monster geben sich im Schloß des Grauens die Türklinken in die Hand.

Sehr gut in der Hand liegt auch die äußerst stabile Verpackung. Die 16-seitige Anleitung erklärt sehr genau, wie die einzelnen Optionen des Spiels handzuhaben sind. Über das "Amiga-Venture" selbst schweigt sie sich eher aus. Sound und graphische Details sind geeignet, dem unvorsichtigen Benutzer so manchen Schauer über den Rücken zu jagen. Vor den Eintritt in



Das Tor zum Schattenreich

SHADOWGATE hat ICOM allerdings die Suche nach dem Schlüssel gesetzt. (Tip: 15 16 05 14 19 03 21 12 12)

Das Ziel ist wie immer kein geringeres als "die Rettung der Welt" - auf geht's.

KINGSOFT

MIKE THE MAGIC DRAGON hat sich seit seinem Auftritt als Demoversion sehr



Der Drache hat es nicht leicht.

zu seinem Vorteil verändert. Es ist ein Spring- und Hüpfspiel, dessen Held, ein kleiner grüner Drache, auf diversen Plateaus verschiedene Gegenstände einsammeln muß. Besonders schön sind auch die Hintergrundgrafiken. Das Thema selbst ist zwar nicht neu, aber Kingsoft präsentiert hier die bisher gelungenste Umsetzung dieses Themas für den Amiga.

Anbieter: KINGSOFT
Grüner Weg 29
5100 Aachen
Preis: 29.95 DM

INERTIA DRIVE

Das Softwarehaus OZISOFT bringt ein Aktionspiel, das vielen vielleicht bekannt vorkommt. Denn auch diese Spielidee hatte bereits einen Vorläufer - SURVIVOR aus dem Jahr 1984 (C64). Der Spieler (nur einer) versucht sich vor auf ihn zustürzenden "Goobers" in Sicherheit zu bringen oder diese abzuschießen, während er gleichzeitig noch etliche Kugeln attackiert, um so die feindliche Festung zur Explosion zu bringen. Die Grafik von Inertia Drive ist ganz ausgezeichnet, die Spielgeschwindigkeit und der Sound stehen dem kaum nach.

VON WOLF DIETRICH

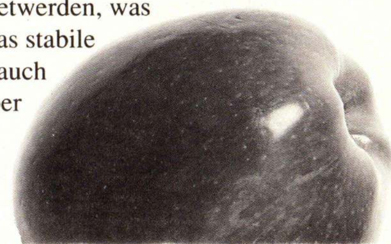
DER GRAFIK JONGLEUR

Die größte Faszination des Amiga liegt sicher nach wie vor im grafischen Bereich. Doch so leistungsfähig diese Maschine ist, man stößt auch hier irgendwann auf Grenzen von Auflösung und Farbvielfalt. Dies mag natürlich auch daran liegen, daß die Ansprüche immer schneller steigen. Für diejenigen, die an diesem Punkt angelangt sind, ergeben sich nun neue Perspektiven, und zwar in Form eines grafischen Subsystems.

GRAFISCHER GEHILFE

Wir hatten dieses (zumindest für den Amiga) neue Produkt bereits im letzen Heft in den News vorgestellt, und nach kurzfristiger Rücksprache mit dem Hersteller, der Firma Sang Computersysteme in Essen, erhielten wir eines der ersten fertigen Geräte zum Test. Hierbei handelte es sich um ein Vorserienmodell, dessen Unterschiede zum endgültigen Modell allerdings nur in der Gehäuseausführung und dem Bus-Stecker bestehen. Die Megavision 02, wie der Name dieses Subsystems lautet, präsentiert sich als eigen-

ständige grafische Einheit. Das Herz des Gerätes ist ein Grafikprozessor vom Typ Hitachi HD63484, welcher auch als ACRTC bezeichnet wird. Weiterhin arbeitet die Megavision mit einem Color LookUp Table (CLUT) vom Typ INMOS G170, welcher dem System eine Farbpalette von 262144 Farben zur Verfügung stellt. Optional läßt sich die Megavision allerdings auch mit einem CLUT INMOS G172 ausrüsten, wodurch die Farbpalette auf 16777216 Farbtöne ausgeweitet wird. Von diesen Farben stehen dann, unabhängig davon, welcher CLUT eingesetzt ist, 256 zur gleichzeitigen Darstellung auf dem Monitor zur Verfügung. Weiterhin ist die Megavision 02 mit einem Megabyte Speicher ausgerüstet, wobei es sich um 256 Kbit-Chips handelt, die 64bitweise organisiert sind. Diese nicht gerade billige Lösung wurde angewandt, da der interne Datenbus des Subsystems 64 Bit breit ist. Angesteuert wird die Megavision 02 über den ExpansionPort des Amiga, also über den Systembus. Dies ist zweifelsohne der schnellste Kommunikationsweg, der sich ohne technische Komplikationen anbietet. Der Busstecker der Megavision ist nicht durchgeschliffen, weshalb das System nur mit solchen Erweiterungen, die ebenfalls auf den Bus gesteckt werden, zusammenarbeitet, die einen durchgeschliffenen Bus haben. Energieprobleme wird man aber mit der Megavision 02 nicht bekommen (wie das bei der Kombination einiger anderer Erweiterungen leider der Fall ist), da sie über ein eigenes Netzteil verfügt. Das Testmodell besaß ein Interface zum Amiga 1000; für die anderen Modelle der Amiga-Serie sind Interfaces in Vorbereitung. Beim Monitorsignal der Megavision handelt es sich um ein RGB-Analog-Signal, das über einen D-Sub-9Pol-Stecker ausgegeben wird. Um die Grafik der Megavision sichtbar zu machen, muß der Anwender über einen Multisync-Monitor verfügen, da sie über die unterschiedlichsten Bildwiedergabefrequenzen verfügt. Der normale Amiga-Monitor ist dabei nicht in der Lage, das Bild zu synchronisieren. Aller-dings ist es sowieso ratsam, über zwei Monitore zu verfügen, da man beim Einsatz der Megavision zwei vollkommen unabhängige Bildsignale hat, und man beim Einsatz von nur einem Monitor ständig umstöpseln müßte, was den Geräten nicht gerade gut tut, abgesehen von der Umständlichkeit einer solchen Vorgehensweise. Die Verarbeitung der Megavision kann durchweg als sehr gut bezeichnetwerden, was sowohl für das stabile Gehäuse als auch die sehr sauber aufgebaute Platine gilt.



Die Bedienungsanleitung der Megavision erweist sich als recht knapp, aber vollständig und gut verständlich. Da es dabei vorallem um die Programmierung des Systems geht, wird sich der versierte Anwender damit ausreichend zurechtfinden. Wer allerdings keine Programmiererfahrung mitbringt und somit mit den Konventionen der C-Programmierung nicht vertraut ist, wird gelegentlich auf Schwierigkeiten stoßen.

BELIEBIGE AUFLÖSUNGEN

Wie auch der Amiga, ist die Megavision 02 nicht auf eine feste Auflösung festgelegt. Allerdings ist die Spanne der Variationsmöglichkeiten beim Subsystem deutlich größer, denn die Auflösung und die Bildwiedergabefrequenz sind hier so miteinander verknüpft, daß sich für den Programmierer vielfältige Möglichkeiten ergeben.

In den mitgelieferten Demoprogrammen, die auch als C-Sources vorliegen, werden erst einmal drei Standardauflösungen verwendet: zum ersten 720*540 Pixel mit 60 Hertz Bildwechselfrequenz, dann 800*600 Pixel mit 55 Hertz Bildwechselfrequenz und zuletzt 1024*768 Pixel mit 30 Hertz Bildwechselfrequenz im Interlace-Modus.



AMIGA 1000

Wie man hierbei sieht, sinkt die Bildwechselfrequenz mit dem Ansteigen der Auflösungen, was in der maximalen Pixelrate von 32 Megahertz begründet liegt, welche der Gesamtzahl der pro Sekunde darstellbaren Pixel entspricht. Bei allen beschriebenen Auflösungen stehen dem Anwender alle 256 gleizeitig einsetzbaren Farben zur Verfügung. Nun ist es allerdings auch möglich, andere Auflösungen zu programmieren, wobei man, aufgrund der sich ändernden Bildwechselfrequenz, eigentlich nur von den Synchronisationsmöglichkeiten des verwendeten Monitors beschränkt wird. Faszinierend ist übrigens, daß eine Auflösung von 1024*768 Pixeln auf einem herkömmlichen Multisyncmonitor dargestellt werden kann, obwohl sie die bei solchen Monitoren angegebene maximale Auflösung bei weitem übersteigt. Wir hatten im Test mehrere verschiedene Monitore dieser Gattung angeschlossen, ohne daß sich Probleme ergaben.

BUNT UND RUND

Diese Überschrift bezieht sich auf die Wiedergabe von Farben und Kreisen. Man kann sie ohne Einschränkung als fantastisch bezeichnen. So ergeben sich beim Einsatz aller Farben Farbverläufe, die in ihren einzelnen Abstufungen nicht mehr erkennbar sind und damit Verläufe im Sinne des Wortes darstellen. Setzt man Pixelmuster aneinander (immer abwechselnd ein Pixel in einer und eines in einer anderen Farbe), so kann man den Unterschied kaum mehr ausmachen, eine so gefüllte Fläche erscheint wie in einer Mischfarbe gefüllt. Dies gilt auch für Linien; so glaubten bei einem Testprogramm, das 1024 vertikale Linien abwechselnd in schwarz und in weiß auf dem Bildschirm darstellt, alle Betrachter, auf eine graue Fläche zu blicken. Auch erscheinen Kreise als wirklich rund; so ist man bei der hohen Interlace-Auflösung (die bei 30 Hertz übrigens schon deutlich flimmerfreier als die des Amiga erscheint) nicht in der Lage, bei Abständen von mehr als einem Meter zum Monitor irgendwelche Ecken in Kreisen auszumachen. Das Lob gilt allerdings nicht nur für Kreise, denn auch die üblichen Computertreppchen bei der Darstellung von schrägen Linien sind kaum mehr zu erkennen. Bei den niedrigeren Auflösungen werden diese zwar schon etwas deutlicher sichtbar, halten sich aber immer noch in Grenzen. Auch Darstellungen von Apfelmännchen können dem schon Amiga-verwöhnten Anwender ehrfürchtiges Staunen entringen, denn hierbei spielt die Megavision ihre Stärken voll aus. Ihre Grenzen sind damit allerdings sicher nicht erreicht, denn die Beschreibung dieser Darstellungen darf man nur als Beispiel für die Fähigkeiten dieses Subsystems sehen. Einige der mitgelieferten Demoprogramme verfehlen übrigens auch in dieser Hinsicht ihre Wirkung nicht: So gibt es ein winziges Programm namens EFFECT, welches nur dazu dient, die Farbpalette in mehreren verschiedenen Schleifen zu verändern; der Effekt (ein Durchlaufen der Farben), der sich gerade bei einem Apfelmännchen ergibt, ist so fantastisch, daß man ihn in Worten nicht beschreiben kann, man muß das einfach gesehen haben.

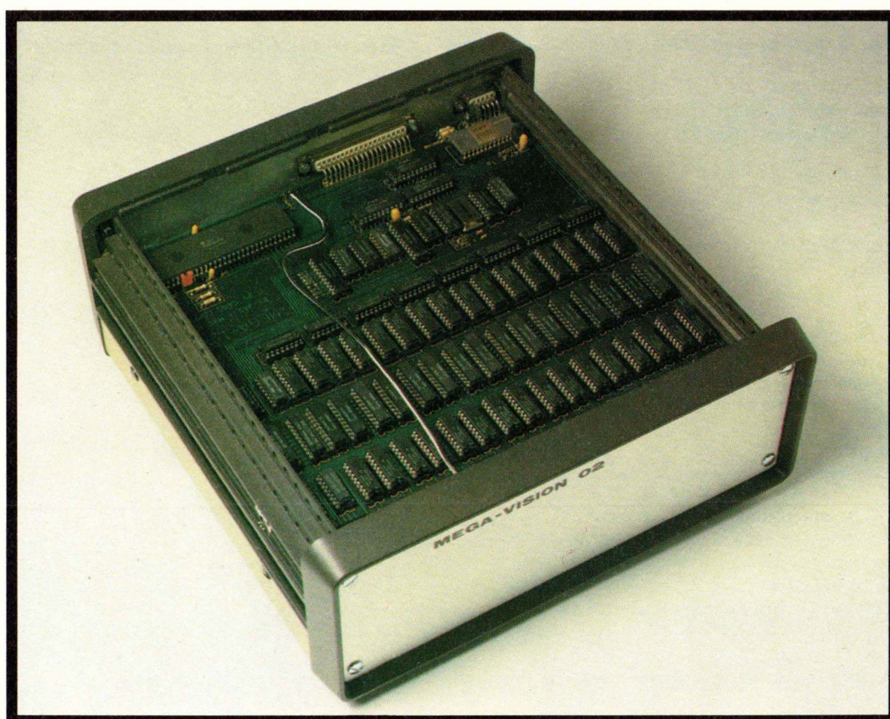
SELBST IST DER PROGRAMMIERER...

Ein heikles Thema, das wohl jedes Computerprodukt betrifft, ist die zur Verfügung stehende Software. Hier

gibt es denn auch, abgesehen von den bereits erwähnten Demos, weiter nichts. Dies bedeutet für den interessierten Anwender, daß er auf seine eigenen Programmierfähigkeiten zurückgreifen muß. Allerdings ist die Megavision 02 sehr leicht zu programmieren, da sie (oder besser: der ACRTC) erstens über sehr wirkungsvolle Befehle verfügt, und zweitens ein Includefile sowie eine Object-Datei zum Einbinden bzw. Einlinken in bestehende C-Programme bereitstehen. Laut Auskunft des Herstellers ist auch eine Include-Datei zum Einbinden in BASIC in Vorbereitung, womit die Befehle zur Steuerung der Megavision dann auch in dieser Sprache zur Verfügung stünden. Weiter oben wurde bereits erwähnt, daß die Megavision mit drei Standardauflösungen arbeitet. Um eine Auflösung von 720*540 Pixeln auf den Bildschirm zu zaubern, muß der Programmierer nichts weiter tun, als einen Befehl namens init() aufzurufen; alles andere erledigt das System in diesem Fall selbst. Übrigens stellt die Megavision dem Anwender in diesem Fall nicht nur einen einzelnen Bildschirm, sondern gleich zwei dieser Sorte und zusätzlich ein Hardwarewindow zur Verfügung. Dies erleichtert das Programmieren von verdeckten Zeichenvorgängen, die erst nach einem Umschalten des Bildschirms sichtbar werden. Das Hardwarewindow läßt sich auf beiden Bildschirmen darstellen; die Einblendungsgeschwindigkeit ist so groß, daß man sie ohne Zeitverzögerungsschleifen kaum mehr wahrnehmen kann, das Window ist bei Aufruf einfach da.

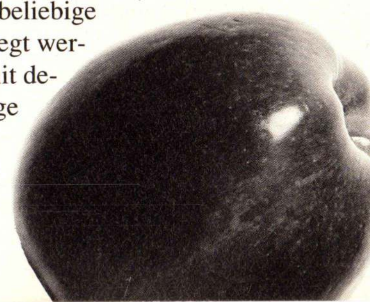
Auch bei höheren Auflösungen stellt die Megavision 02 dem Anwender noch zwei Screens und ein Window zur Verfügung; letzteres wird allerdings in der möglichen Maximalgröße immer weiter eingeschränkt, da der Speicherplatzbedarf der Screens steigt. Bei 1024*768 Pixeln ist aber auch Schluß mit dem zweiten Bildschirm, da der erste Screen in diesem Fall nach Adam Riese bereits 768 Kbyte belegt. Will man andere als die über init() abrufbaren Auflösungen festlegen, so bietet der ACRTC umfangreiche Programmierungsmöglichkeiten in Bezug auf die Register, die für die Bilddarstellung verantwortlich sind. Man kann das Synchronisationssignal so an die verschiedensten Monitore anpassen, daß das Bild stets unverzerrt und sauber wiedergegeben wird. Auch wenn man nicht über die genauen Synchronisationswerte des Monitors verfügt, kommt man hier mit ein bißchen Ausprobieren zum Ziel. Andere Register legen Auflösung und Bildursprung fest, wobei man den Bildursprung zum Beispiel nach links oben oder unten oder auch in der Bildmitte festlegen kann. Alle Register lassen sich auch über einen einfachen Befehl auslesen. Der ACRTC stellt dem Programmierer einige Befehle zur Verfügung, mit denen man direkten Einfluß auf den Bildwiederholspeicher nehmen kann. So ist es beispielsweise kein Problem, bestimmte Werte in bestimmte Speicherbereiche zu schreiben oder daraus zu lesen. Dies erweist sich übrigens als einfacher als beim Amiga, da der Bildschirmspeicher der Megavision 02 nicht in Form von mehreren Bitmaps, sondern linear organisiert ist, was von der Prozessorarchitektur des ACRTC verlangt wird.

Wer nun glaubt, die Programmierung der Megavision laufe auf das Schreiben von Werten in Register und in bestimmte Speicherbereiche hinaus, der sei hiermit beruhigt: Auch einfache Grafikbefehle existieren in großer Auswahl. Linien, Kreise, Ellipsen, Kreis- und Ellipsenbögen lassen sich mit einfachsten Befehlsaufrufen darstellen. Natürlich lassen sich Flächen auch füllen. Weiterhin verfügt der ACRTC über einen Befehl, der rechteckige Bildschirmausschnitte kopiert und damit Amiga-eigenen Befehlen entspricht. Dieser Befehl, agcpy() genannt, ermöglicht sogar das Drehen dieser Bildauschnitte in 90-Grad-Schritten. Auch das Setzen der Farben erweist sich als problemlos; wie der Amiga kennt die Megavision primäre und sekundäre Zeichenfarbe, und die Definition einer Farbe unterscheidet sich eigentlich nur dahingehend von der Festlegung beim Amiga, daß bei der Megavision jeder Farbanteil (für Rot, Grün und Blau) in 64 Schritten einstellbar ist.



Die Europlatine

Damit ist es aber noch nicht genug: Der ACRTC kann beim Kopieren von Bildausschnitten beispielsweise logische Verknüpfungen zwischen Quelle und Zielbereich vornehmen; verschiedene Modi lassen dem Anwender hier einige Möglichkeiten offen. Weiterhin arbeitet der ACRTC mit einem Pattern-RAM, in dem beliebige Muster festgelegt werden können, mit denen rechteckige Bereiche oder aber auch



beliebige Flächen gefüllt werden können. Auch ein Clipping, das das Zeichnen inner- oder außerhalb eines festgelegten Ausschnitts unterdrückt, kann über einen einfachen Befehl aktiviert werden. Diese Aufzählung von Möglichkeiten, die der ACRTC dem Programmierer bietet, ist beileibe nicht vollständig, was auch für die Liste der Befehle gilt, die diesem Artikel beigefügt ist. Es würde wohl auf einige Seiten hinauslaufen, alle Befehle dieses Systems auch nur kurz zu behandeln; diese Liste soll nur eine ungefähre Vorstellung der Möglichkeiten, die die Megavision 02 bietet, vermitteln. Ein kleines Listing in C, welches diesem Artikel beigefügt wurde, demonstriert, wie einfach die Programmierung des Subsystems in der Praxis ist. Hierbei werden allerdings auch nur ein paar der einfacheren Befehle aufgerufen.

THEORIE UND PRAXIS

In Anbetracht der oben geschilderten Möglichkeiten stellt sich die Frage: Wie aufwendig ist die Programmierung der Megavision wirklich? Hierzu läßt sich eigentlich nur sagen, daß jeder, der in der Lage ist, seinen Amiga zu programmieren, und laufe es nur darauf hinaus, Kreise und Linien auf dem Bildschirm zu zeichnen, dies auch mit der Megavision kann. Ein kleines Mandelbrot-Programm, das fertig für den Amiga bereitlag, ließ sich innerhalb von fünf Minuten an die Megavision anpassen, und das Resultat war mehr als überzeugend. Etwas mehr Arbeit erforderte die Anpassung des Moviemakers, den wir in Heft 11/87 abgedruckt hatten; diese Arbeit lief allerdings im wesentlichen darauf hinaus, für jede Fläche immer einen Punkt zu finden, der innerhalb der Fläche liegt und als Ansatzpunkt für den Füllbefehl des ACRTC verwendet werden kann. Alles andere nahm ebenfalls nur kurze Zeit in Anspruch.

Nun stellt sich noch die Frage, wie es mit der Arbeitsgeschwindigkeit der Megavision aussieht. Dies läßt sich allerdings nicht einfach beantworten, da die verschiedensten grafischen Operationen auch verschiedene Zeiten beanspruchen. Generell kann man sagen, daß die Megavision sehr schnell ist. Bei einigen Operationen schlägt sie hierbei den Amiga um Längen. So lief ein Programm, das einhundertmal das Window in den Vordergrund und wieder in den Hintergrund schaltete, ohne Verzögerungsschleifen etwa eine halbe Sekunde; hier kam bereits der Monitor nicht mehr mit. Vom Hersteller werden für das Zeichnen von Punkten und Linien bis zu 2 Millionen Pixel pro Sekunde angegeben, für das Kopieren von Bildschirmbereichen bis zu 4 Millionen Pixel/sec. und für das Füllen (rechteckiger Flächen) bis zu 8 Millionen Pixel/sec. Leider ließen sich solche Werte mit einfachen Testprogrammen nicht erreichen; so erreichte ein Programm mit einer in einer Schleife durchgeführten Fülloperation (Rechtecke) eine ungefähre Füllrate von 1.2 Millionen Pixel pro Sekunde. Natürlich kann in diesem Fall mittels Optimierung noch ein Zeitgewinn herausgeholt werden, aber die optimistischen Vorgaben zu erreichen, erscheint doch als relativ utopisch, vor allem in Anbetracht der Tatsache, daß die Leistungsfähigkeit des ACRTC vom Hersteller (Hitachi) selbst nicht so hoch angesetzt wird wie vom

```
Einige Befehle der Megavision 02 bzw. des ACRTC

init();
Eröffnet zwei Bildschirme in der Auflösung 720*540 Pixel, dazu ein Hardwarewindow.

disp screen(BASE);
Zeigen des ersten Bildschirms. Kann auch für den zweiten Bildschirm eingesetzt werden
(UPPER).

clear screen(screen,color);
Löschen eines Bildschirms in beliebiger Farbe.

amove(x,y);
Positionieren des Grafikcursor auf der angegebenen Bildschirmposition.

aline(mode,x,y);
Linie ziehen.

dot(mode);
Punkt zeichnen auf der Position des Grafikcursors.

arct(mode,x,y);
Rechteck vom Grafikcursor zu den angegebenen Koordinaten zeichnen.

crcl(mode,radius);
Kreis an der Position des Grafikcursors zeichnen.

elps(mode,a,b,dx);
Ellipse mit hor. Radius dx und Verhältnis a:b zeichnen.

afrct(mode,x,y);
Wie arct(), füllt das Rechteck aber aus.

apll(mode,n,liste);
Polylinie ziehen. n entspricht der Anzahl Punkten, liste ist das Koordinatenarray.

( Alle Zeichenbefehle, die mit einem "a" beginnen ( für ABSOLUT ), gibt es auch mit einem
"r" ( für RELATIV ).

paint(mode);
Füllen vom Grafikcursor zur Randfarbe.

set color0(n);
set color1(n)
set coloredge(n)
Primäre und sekundäre Zeichenfarbe, Randfarbe auswählen.

regwrite(nummer,inhalt);
Werte in ein Displayregister schreiben.

regread(nummer);
Register auslesen.

select color(wert);
Farbe zum Verändern anwählen.

set color(r,g,b);
Farbe definieren.

syncwait();
Auf vertikale Synchronisation warten.

Wie erwähnt, ist das nur ein Auszug aus der Liste der Grafikbefehle des ACRTC. Für den
geübten Programmierer gibt es noch eine Menge weiterer Möglichkeiten, den ACRTC zu
beeinflussen.
```

KiCK START

KiCK START

KiCK START

KiCK START

KICKSTART-GRUßKARTE

Bitte
frei-
machen



DIE FACHZEITSCHRIFT FÜR
DEN AMIGA-ANWENDER

KICKSTART-GRUßKARTE

Bitte
frei-
machen



DIE FACHZEITSCHRIFT FÜR
DEN AMIGA-ANWENDER

KICKSTART-GRUßKARTE

Bitte
frei-
machen



DIE FACHZEITSCHRIFT FÜR
DEN AMIGA-ANWENDER

´TOP 12´

Mein Lieblingsspiel

**Wettbewerbsbedingungen siehe
TOP 12 in diesem Heft**

Bitte
frei-
machen

*´MERLIN´- Computer GmbH
Redaktion KICKSTART
´TOP 12´
Industriestraße 26*

D-6236 Eschborn

Hersteller der Megavision. Nichtsdestotrotz sind die ermittelten Geschwindigkeiten sehr hoch; das gilt auch für das Füllen beliebiger Flächen (von Hitachi übrigens mit einer Rate von 0.5 Megapixeln pro Sekunde angegeben), das zwar etwas langsamer ist als beim Amiga selbst, aber dennoch schnell genug vonstatten geht, um einfachere dreidimensionale Objekte zu animieren. Man sollte bei all diesen Vergleichen aber auch berücksichtigen, daß die angegebenen Geschwindigkeiten immer für die 256 simultan verfügbaren Farben gelten; beim Amiga kann man bei einer Erhöhung der Farbanzahl stets eine Verringerung der Zeichengeschwindigkeit feststellen.

EINSATZBEREICHE

Die Frage nach diesen muß man wohl bei jedem Computerprodukt stellen. Für die Megavision 02 lautet die Antwort einfach, aber universell: Dieses Subsystem ist für praktisch jede Art der grafischen Anwendung hervorragend geeignet. Natürlich erweist es sich als Manko, daß für dieses System keine vorgefertigte Software zur Verfügung steht, aber wer sich die Programmierung spezifischer Anwendungen auf dem Amiga zutraut, der darf sich dies auch bei der Megavision zutrauen. Natürlich orientiert sich ein solches System an professionelleren Maßstäben, und dies kann es sich durchaus leisten. Mit einer solchen Erweiterung wird der Amiga zu einer Art Micro-Workstation, die den Leistungen einer Sun- oder einer Apollo-Workstation zumindest in der Grafikdarstellung schon recht nahe kommt. Wer sich zutraut, CAD-Anwendungen oder gar Ray Tracing-Module zu programmieren, kann dieses System durchaus professionell nutzen. Ein Programm mit einer Zielsetzung wie beispielsweise Videoscape 3D würde dem Anwender auf einem solchen System neue Bereiche erschließen. Dieses Beispiel wurde gewählt, weil Videoscape trotz Beschränkung auf sechzehn Farben fantastische Darstellungen auf den Bildschirm zaubert;

```
/* Demoprogramm zum Megavision 02 Grafiksubsystem by WW */
/********************************************************/
/* Zur Einbindung der Megavision-Befehle und Predefines */
#include <acstuff.h>

main()
{

 /* Screens 720*540 oeffnen */
 init();
 /* Cursor auf Bilschirmmittelpunkt */
 amove(360,270);
 /* Farbe 255 als Zeichenfarbe setzen */
 set_color0(255);
 /* Einen Kreis mit Durchmesser 100 Pixel zeichnen */
 crcl(0,100);
 /* Nun ein gefuelltes Rechteck, 100^2 Pixel */
 set_color0(155);
 amove(100,100);
 rfrct(0,100,100);
 /* Ein Kreisbogen */
 set_color0(55);
 amove(500,400);
 aarc(0,550,450,600,500);
 /* Das wars schon. Ist so einfach, wie es aussieht */
}
```

gäbe man einem Programm, das die Megavision unterstützt, die gleichen Möglichkeiten mit auf den Weg (wie zum Beispiel das Darstellen von Farben, die nicht in der Farbpalette existieren, mittels pixelweisem Mischen anderer Farben, was bei der Auflösung der Megavision nicht mehr als gerastert wahrgenommen werden kann), so stieße man in wirklich professionelle Bereiche vor, in denen beispielsweise die Produktion von Low-Cost-Computeranimationen auf Video möglich wäre. Dies bleibt natürlich solange Spekulation, solange die dafür notwendige Software nicht zur Verfügung steht.

Als "Spielzeug" für grafikinteressierte Anwender erscheint die Megavision 02 wohl zu teuer, was auch in Anbetracht der Tatsache gilt, daß man zum Kauf eines zweiten Monitors gezwungen ist (der auch von besserer Qualität sein muß als der Amiga-Monitor), aber für denjenigen, der sich ernsthaft mit Computergrafik beschäftigt und auch die Fähigkeiten und die Motivation mitbringt, sich Anwendungen selbst zu programmieren, läßt sich dieses System uneingeschränkt empfehlen. Und wer einmal selbst (wie der Autor) in den Genuß der wirklich fantastischen Grafik, des Arbeitens auf zwei Bildschirmen und des Workstation-Feelings gekommen ist, dem wird sein Amiga unter Umständen in Zukunft wie ein halbes Gerät vorkommen.

Megavision 02

Preis: DM 2740.-
incl. Interface und Netzteil

Hersteller:
Sang Computersysteme Gmbh
Am Wünnesberg 13
4300 Essen


MSDOS

VON CHRISTIAN KELLER

PC INTIM

TEIL 1: Die Grafik des PC

Für den PC erhält man eine große Anzahl verschiedener Grafikkarten, die sich alle mehr oder weniger in der Auflösung und der Farbenzahl unterscheiden. In diesem Beitrag soll eine Übersicht der verschiedenen Adapter gegeben werden. Auf die Bildmodi, die vom AMIGA emuliert werden können, soll dabei besonderes Augenmerk gelegt werden.

Ist alles so schön bunt hier

Als der IBM-PC auf den Markt gebracht wurde, war er nur mit einer nicht grafikfähigen Videokarte bestückt. Da dies den meisten Anwendern nicht genügte, kamen nach und nach diverse Grafikkarten auf den Markt, die verschiedene Auflösungen und Farbmöglichkeiten zu bieten haben. Von den etablierten Grafikmodi stammt nur einer nicht von IBM selbst, die Hercules-Grafik. Alle anderen Grafikkarten wurden zuerst von Big Blue kreiert und dann von vielen Herstellern abgekupfert. Durch den dadurch entstandenen Konkurrenzkampf werden heute alle Grafikkarten zu einem äußerst günstigen Preis angeboten, was eine Nachrüstung der Sidecar oder des Bridgeboard überlegenswert erscheinen läßt.

Um einen Überblick zu gewinnen, sollen hier erst einmal alle Grafikkarten dargestellt werden.

1. Der Monochrom-Display-Adapter (MDA).

Diese Karte ist zur Wiedergabe eines 80 x 25 Zeichen großen monochromen Bildschirms fähig. Grafik wird nicht unterstützt. Die Darstellung eines Zeichens erfolgt in einer 9 x 14 Punktmatrix. Die Charaktere selbst sind dabei 7 x 9 Punkte groß. Dadurch ist eine gute Darstellung von Zeichen mit Unterlängen und Unterstreichungen gewährleistet. Folgende Zeichenattribute sind möglich: normal, leuchtend, blinkend, unterstrichen und unsichtbar. Die Bildwiederholrate beträgt 50 Hertz. Die Zeichendarstellung kann sehr gut genannt werden.

Das Manko dieser Video-Karte ist die fehlende Grafikfähigkeit. Diese Karte kann (mit Einschränkungen) vom AMIGA emuliert werden.

2. Das Color-Grafik-Adapter (CGA)

Mit einem CGA sind sechs verschiedene Darstellungsweisen möglich: 4 Textmodi und 2 Grafikmodi. Die Textmodi sind die folgenden:

40 x 25 Zeichen schwarzweiß
80 x 25 schwarzweiß
40 x 25 in 16 Farben
80 x 25 (16 Farben)

Dazu gesellen sich die Grafikauflösungen 320 x 200 Punkte in vier Farben und 640 x 200 Pixels monochrom. Die Darstellung der Textzeichen erfolgt mit einer 8 x 8 Punktmatrix, was eine erhebliche Qualitätsminderung gegenüber der MDA-Karte bedeutet. Die Bildwiederholrate beträgt 60 Hz. Neben den 16 Farben werden auch blinkende Zeichen unterstützt. Auch diese Grafikkarte wird vom Amiga emuliert.

3. Die Hercules-Karte (auch MGA genannt)

Die Fähigkeiten dieser Karte ermöglichen die gute Zeichendarstellung des MDA, ohne ganz auf Grafik verzichten zu müssen. Im Textmodus ist die Hercules-Karte zum MDA kompatibel. Daneben wird noch ein Grafik-Modus geboten, der eine Auflösung von 720 x 348 Punkten erlaubt. Die Grafik ist dabei nur monochrom möglich.

Grafikk.	Adressraum
MDA	B0000h - B0FFFh
CGA	B8000h - BBFFFh
MGA	B0000h - BFFFFh
EGA	A0000h - BFFFFh

Abb.1 : Speicherbelegung der gebräuchlichsten Grafikkarten

4. Das Enhanced Graphics Adapter (EGA)

In der Anfangszeit durch ihren hohen Preis und den noch höheren Preis des benötigten, hochwertigen Farbmonitors kaum beachtet, mausert sich die EGA-Karte wegen der stark gefallenen Hardwarepreise in letzter Zeit zum Renner. Alle Betriebsarten der MDA- und CGA-Karte werden unterstützt. Zusätzlich bedient die EGA-Karte noch folgende Text- und Grafik-Auflösungen:

80 x 25 Zeichen in 16 (von 64 möglichen) Farben (Zeichenmatrix: 8 x 14 Punkte)

640 x 200 Bildpunkte in 16 Farben

640 x 350 Bildpunkte monochrom.

Die Bildwiederholrate beträgt 60 Hz.

Neben diesen Standard-Grafikkarten gibt es noch eine Reihe von anderen Videoadaptern, die sich nicht auf dem breiten Markt durchsetzen konnten, da sie meist für besondere Anwendungen (wie z.B. CAD) ausgelegt sind. Die Auflösung reicht hier bis zu 1024 x 1024 Bildpunkte.

Eine Sonderstellung nehmen die Multifunktionskarten ein. Auf diesen Adaptern sind verschiedene Grafikkarten zusammengefaßt. Als Beispiel hierfür sei nur die AGA-Karte von Commodore genannt: CGA-, MDA- und Herculeskarte werden voll ersetzt. Daneben bietet die AGA-Karte noch eine Plantronics-Colorplus-Adapter-Emulation. In diesem Modus können 132 Spalten Text dargestellt werden oder eine Grafik von 640 x 200 Punkten in 16 Farben.

Vortäuschung falscher Tatsachen

In der obigen Tabelle wurde gesagt, daß die MDA-Karte nur mit Einschränkungen emuliert wird. Diese Unzulänglichkeit beruht auf dem Bildschirmformat des AMIGA. Der Amiga hat eine grafikorientierte Bilddarstellung. Text stellt also nur eine Abart von Grafik dar. Die Grafikauflösung beträgt 640 x 200 (bzw. 256) Punkte. Berechnet man die Anzahl der Bildpunkte beim MD-Adapter, erhält man eine wesentlich höhere Auflösung. 80 x 25 Zeichen in einer 9 x 14 Zeichenmatrix ergeben nämlich 720 x 350 Punkte. Diese Auflösung erreicht der AMIGA nicht (720 quer schafft er nie, und die Umschaltung in den Interlacemodus - in welchem der AMIGA 350 Punkte in Y-Richtung darstellen könnte - blieb uns gottlob erspart.) Aus diesem Grunde werden alle Zeichen im Monochrommodus in einer 8 x 8-Matrix dargestellt. Dies entspricht der Wiedergabe im CGA-Modus. Die wesentlich bessere Darstellung des MDA hat man auf dem AMIGA also nicht. Von der Ansteuerung her ist die Emulation mit der MDA-Karte gleich. Speicherbereich, Portadressen und Register des Video-Controllers entsprechen sich.

Ein weiteres Manko ist die fehlende Implementation der blinkenden Zeichendarstellung. Bei den IBM-Grafikkarten wird das Blinken durch die Hardware des Video-Adapters bewerkstelligt. Der AMIGA müßte dies bei der Vortäuschung der Karte mit (wertvoller) CPU-Rechenzeit erzeugen. Da die Option Blinken nur von geringer Bedeutung ist, haben die Entwickler der Sidecar auf dieses Feature verzichtet. Dem Blinken des Lichtmarkenzeigers (für die Freunde der zwanghaft eingedeutschten Computerterminologie) wollte man aber nicht entsagen. Um den Cursor rhytmisch aufleuchten zu lassen, muß sich daher der 68000'er des öfteren bemühen. Um das Ganze noch schöner zu gestalten, kann man sogar zwischen vier Blinkraten wählen. Normale PCs können nur zwei Geschwindigkeiten ausgeben.

Wir bauen uns ein Bild

Jetzt sollen die vom AMIGA emulierten Grafikkarten einmal etwas näher unter die Lupe genommen werden. Dabei geht es hauptsächlich um die Art und Weise, wie und wo das Bild auf einem PC abgelegt wird. Auf was hier nicht eingegangen wird (und wovor Leute ohne fundierte Hardwarekenntnisse und ohne entsprechende Unterlagen nur gewarnt werden können), sind direkte Manipulationen der Video-Controller im I/O-Bereich. MS-DOS bietet für solche Angelegenheiten einige Routinen, die man tunlichst benutzen

Funktion	Blink	Hintergr.	Intens.	Vordergr.
Bit	7	6 5 4	3	2 1 0
Keine Anzeige	B	0 0 0	I	0 0 0
Unterstrichen	B	0 0 0	I	0 0 1
Normale Darst.	B	0 0 0	I	1 1 1
Inverse Darst.	B	1 1 1	I	0 0 0
B: Blinken an	1			
I: Zeichenint.			1	

Abb.2 : Belegung des Attribut-Bytes bei der MDA-Karte

Funktion	Hintergr.	Intens.	Vordergr
Bit	6 5 4	3	2 1 0
Schwarz	0 0 0	0	0 0 0
Rot	1 0 0	0	1 0 0
Grün	0 1 0	0	0 1 0
Blau	0 0 1	0	0 0 1
Braun	1 1 0	0	1 1 0
Violett	1 0 1	0	1 0 1
Türkis	0 1 1	0	0 1 1
Hellgrau	1 1 1	0	1 1 1
Dunkelgrau	n.d.	1	0 0 0
Hellrot	n.d.	1	1 0 0
Hellgrün	n.d.	1	0 1 0
Hellblau	n.d.	1	0 0 1
Gelb	n.d.	1	1 1 0
Lila	n.d.	1	1 0 1
Helltürkis	n.d.	1	0 1 1
Weiß	n.d.	1	1 1 1

Zusätzlich kann Bit 7 für blinkende Zeichen gesetzt werden.
n.d. = nicht darstellbar

Abb. 3 :Attribute der CGA im Farbmodus

sollte. Auf die Benutzung dieser Routinen wird in einem der nächsten Hefte eingegangen, da hierzu einige Vorbemerkungen vonnöten sind (Interruptprogrammierung usw.).
Wie oben schon angedeutet, hat der PC im Gegensatz zum AMIGA keinen grafikorientierten Bildschirm. Beim Text-Bildschirm des PCs werden nur die ASCII-Zeichen (und die zugehörigen Attribute) gespeichert. Aus diesen Informationen erstellt der Video-Prozessor dann die einzelnen Zeichen. Die Form der Zeichen ist hierfür in einem EPROM abgespeichert, kann also nicht verändert werden. Ein Verwenden verschiedener Fonts ist deswegen nicht möglich. Dieses Vorgehen hat aber den Vorteil des geringen Speicherbedarfs. Im Textmodus wird grundsätzlich ein Wort (16 Bit) zur Abspeicherung eines Zeichens benötigt. Im Byte mit der geraden Adresse (Even Byte) ist die Information über das darzustellende Zeichen abgelegt. Der Wert entspricht dem ASCII-Wert des Charakters. Im eine Adresse höher liegenden Byte (Odd Byte) sind die Informationen über die Darstellungsweise gespeichert. Diese Informationen sind je nach Grafikadapter unterschiedlich. Jedes Bit schaltet sozusagen eine "Besonderheit" ein. Die Bedeutungen der Bits im Odd Byte sind in Abb.2 (für MDA) und Abb.3 (für CGA) dargestellt. Auf diese Art und Weise benötigt man für einen 80 x 25 Zeichenbildschirm 80 x 25 x 2 = 4000 Bytes. Dies entspricht auch (fast) genau dem Adreßraum der MDA-Karte. Die CGA-Karte bietet aber etwa viermal soviel Platz (der später auch ganz für die Grafik benutzt wird). Um diesen Platz auch im Textmodus zu nutzen, hat man in der CGA-Karte vier Textseiten hintereinander gelegt. Diese Seiten können unabhängig programmiert werden. Die Verteilung der Seiten ist in Abb.4 ersichtlich. Für den 40-Zeichen-Modus braucht man nur die Hälfte des Speichers pro Seite, man kann also 8 Seiten implementieren. Daraus ergeben sich für die einzelnen Zeichen folgende Speicherstellen im PC-Adreßraum:

MDA:
B0000h +(Zeile-1)*A0h+(Spalte-1)* 2

CGA:
40 Zeichen:
B8000h+Seite * 800h + (Zeile-1) * 50h + (Spalte-1) * 2

80 Zeichen:
B8000h + Seite * 1000h + (Zeile-1) * A0h + (Spalte-1) * 2

Die Adressen, die mit diesen Formeln errechnet werden, geben die Adresse des Wortes für das Zeichen mit entsprechender Zeilen- und Spaltenposition. Für Byte-weise Manipulation spezifiziert der Wert die Charakter-Adresse, für die Attributadresse muß man 1 dazuaddieren.
Bezüglich des Colormodus ist noch einiges zu den Farbdarstellungen zu sagen. 16 Farben stehen nämlich nur für den Vordergrund zur Verfügung, das Intensitätsbit wirkt nur auf den Vordergrund (also das Zeichen). Zur Ansteuerung des Hintergrundes stehen nur die drei Grundfarben zum Mischen zur Verfügung (2hoch3, also 8 Farben). Ebenso kann der PC normalerweise nur 16 fest eingestellte Farben darstellen. Das liegt an dem Tatbestand, daß mit den Farbbits nur die Elektronenstrahlen ein- und ausgeschaltet werden (bzw. mit dem Intensitätsbit der Strahlstrom erhöht wird). Daß bei der Emulation durch den AMIGA die Farben frei wählbar sind, beruht auf der Umsetzung des Bildes ins AMIGA-Format.

40 x 25 Seite	Adressen	80 x 25 Seite	Adressen
0	B8000h - B87CFh	0	B8000h - B8F9Fh
1	B8800h - B8FCFh		
2	B9000h - B97CFh	1	B9000h - B9FCFh
3	B9800h - B9FCFh		
4	BA000h - BA7CFh	2	BA000h - BAFCFh
5	BA800h - BAFCFh		
6	BB000h - BB7CFh	3	BB000h - BBFCFh
7	BB800h - BBFCFh		

Abb.4: Seitenverteilung der CGA-Karte in den verschiedenen Textmodi

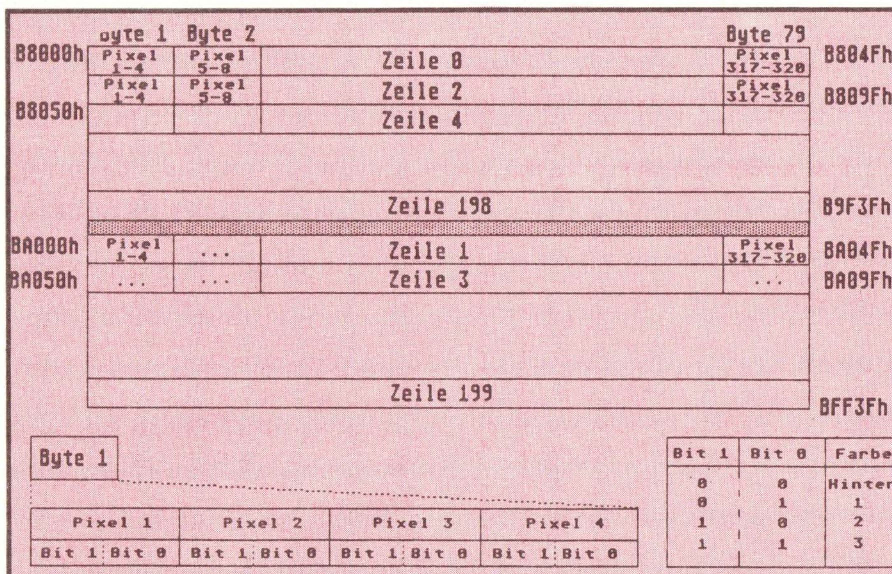


Abb.5: Speicherbelegung beim 320x200 Modus

In den Grafikmodi der CGA-Karte sieht die Speicherverteilung ganz anders aus. Im LoRes-Modus(320 x 200 Punkte in vier Farben) werden pro Pixel nur noch 2 Bits benötigt. In einem Byte haben demnach 4 Pixel Platz. Auch hier werden die Informationen spaltenweise abgelegt. Im Byte B8000h stehen so 1., 2., 3. und 4. Pixel, in Byte B8001h 5., 6. ...usw. Das 320. Pixel belegt die beiden letzten Bits von B804Fh. Und dann? Dann geht's bei BA000h weiter. Der PC speichert im Grafikmodus zuerst alle Spalten mit gerader Nummer, danach beginnt er mit der Speicherung der ungeraden Nummern. Das ist hardwarebedingt, um Textdarstellung und Grafik zu ermöglichen. Im Lo-Res-Modus stehen zwei Farbpaletten zur Verfügung, bei denen Hintergrundfarbe und Intensität wählbar sind. Im HiRes-Modus (640 x 200 Pixel monochrom) werden dann in jedem Byte acht Pixel abgelegt. Wie die genaue Speicherbelegung in den Grafikmodi ist, können sie in Abb.5 und 6 sehen.

Eine Grafikkarte soll her

Wem die Möglichkeiten der Videokarten-Emulation nicht ausreichen, hat natürlich die Möglichkeit, seinen Rechner mit einer gesonderten Grafikkarte auszurüsten. Dabei gibt es aber einiges zu beachten:

1. Die zusätzliche Grafikkarte kann ihr Bild nicht über den Amiga ausgeben. Die Karte braucht einen eigenen Monitor, der auch zur Grafikkarte passen muß. Für die monochromen Video-Adapter genügt ein einfacher Grün- (oder Gelb-) Monitor. Gegenüber der Emulation läßt sich eine starke Verbesserung der Textdarstellung erreichen. Zu einer CGA-Karte benötigt man einen Farbmonitor mit RGBI-Eingang für TTL-Pegel. Der AMIGA-Monitor besitzt solch einen Eingang, jedoch ist es nicht ratsam, während des Betriebes den Monitor umzustecken. Im günstigsten Fall ist es sehr unpraktisch, im schlimmsten Fall kann die Hardware beider Rechner Schaden nehmen.
Für eine EGA-Karte ist ein hochwertiger Farbmonitor vonnöten. Man hat hier die Wahl zwischen einem speziellen EGA-Monitor oder den sogenannten Multifrequenz-Bildschirmen. EGA-Monitore sind speziell auf die EGA-Karte abgestimmt. Mit anderen Grafikkarten sind sie nicht betriebsfähig. Die Multifrequenz-Monitore hingegen können an verschiedenen Grafikkarten betrieben werden. Eine Benutzung am AMIGA ist so auch möglich. Da die Preisunterschiede zwischen beiden Monitortypen nicht so gravierend sind (EGA: ca. 1100 DM, Multifr.: ca. 1500 DM), ist dem Multi meines Erachtens eindeutig der Vorrang zu geben.

2. Da die Speicherbereiche der Grafikkarten sich teilweise überschneiden, kann es notwendig sein, die Grafikemulationen teilweise oder ganz abzuschalten. Bei Verwendung einer MDA- oder CGA-Karte muß nur die jeweilige Emulation außer Kraft gesetzt werden. Beim Einsatz von Herculesgrafik oder EGA-Karte müssen beide Grafikumsetzungen über den Amiga ausgeschaltet sein. Beachten Sie, daß bei komplett ausgeschalteter Emulation keine Kontrolle des PCs über den AMIGA-Monitor mehr möglich ist. Die Abschaltung der Emulationen erfolgt bei der Sidecar über die DIP-Schalter 1 und 2, beim Bridgeboard über die Jumper J1. Beim A2000 mit Bridgeboard ist die Wahl des Videomodus auch über PCPrefs möglich, eine Unverträglichkeit dieser softwaremäßigen Umschaltung in besonderen Fällen läßt sich jedoch nicht mit letzter Sicherheit ausschließen.

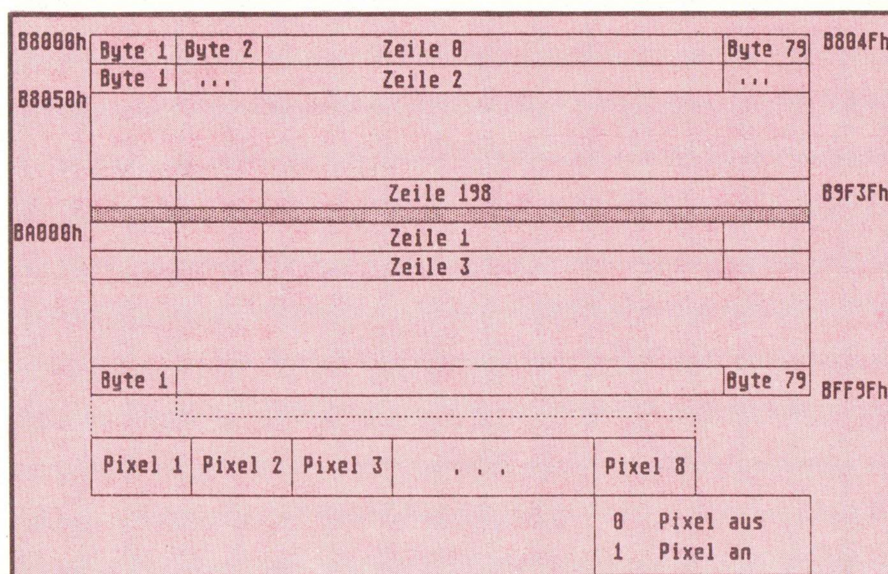


Abb.6: Die 640x200 Punkte - Grafik

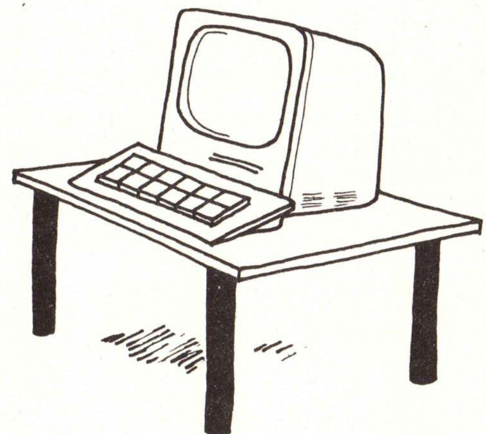
3. Nur bei der EGA-Karte kommt noch eine weitere Besonderheit hinzu. Da die EGA-Grafik auch den Bereich A0000h-AFFFFh belegt, darf das Filetransfer-Segment des Dual-Ported-Rams nicht in diesem Bereich liegen. Da man drei Segmente zur Auswahl hat, dürften hier keine Schwierigkeiten auftreten. Bei der Sidecar wird das Segment wieder über Dipschalter ausgewählt. Beim Betrieb einer EGA-Karte muß Schalter 4 auf OFF stehen. Das Bridgeboard erlaubt eine Einstellung nur über PCPrefs. Hier muß man natürlich darauf achten, daß man das Segment während des Betriebes nicht versehentlich verstellt.

Im nächsten Teil wird die genaue Ansteuerung des MDA und der CGA-Karte beschrieben.

Literaturverzeichnis:
Technisches Handbuch PC 10/20
Commodore



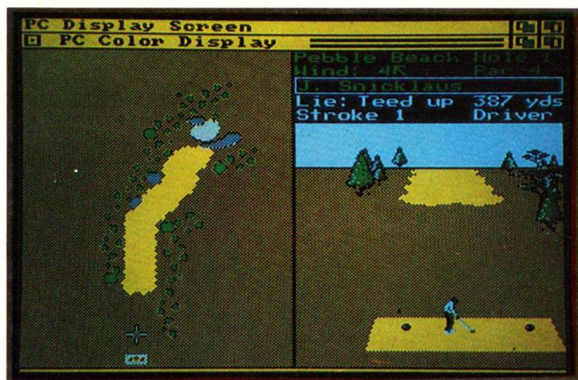




VON VOLKER SEHORZ

MSDOS

PC GAMES



WORLD TOUR GOLF

Dem Spiel beigelegt ist ein 12 - seitiges Kompendium, das alles Wissenswerte über Golf, und wie man es spielt beinhaltet. Hier wird beispielsweise beschrieben, welcher Schläger wann zu benutzen ist oder welche Schlagstärke bei welcher Entfernung erforderlich ist. Es klärt außerdem über Fachbegriffe wie 'Par', 'Double Bogie', 'Albatross' und 'Chipping' auf. Da das Spiel ohne Kopierschutz geliefert wurde, empfiehlt es sich, wie im Heft beschrieben, das Spiel von einer Festplatte aus zu betreiben. Eine Art Kopierschutz befindet sich dennoch im Spiel, d.h. am Anfang wird man aufgefordert, einen von 12 im Beiheft beschriebenen Plätzen, durch ein kleines Bild auf dem Monitor dargestellt, mit Namen zu benennen. Das Spiel funktioniert auch dann, wenn man nichts eingibt und nur die Return - Taste drückt, aber nachfolgend sind pro Spielplatz nur noch 2 Löcher vorhanden.

Golf läßt sich mit 1-4 Spielern und, wenn man es wünscht, auch mit dem Computer spielen. Unterstützt wird neben der Tastatur auch der Joystick - Betrieb, wobei man dazusagen muß, daß die Tastatur unabdingbar ist. Die Hardcopy-Option bleibt während des gesamten Spieles erhalten. Insgesamt stehen 20 Plätze und 14 Schläger zur Verfügung. Mehrere Schwierigkeitsgrade (Windstärken, Untergründe, Wasserflächen und Geländeerhebungen) machen das Spiel sehr vielseitig. Schlagrichtung, Schlagstärke und Schläger lassen sich individuell auf jedes einzelne Loch einstellen. Nach jedem Einlochen erscheint die Punktetabelle und gibt Auskunft über die erbrachte Leistung.

Zu Anfang des Spiels kommt man ins Aktivitäten-Menü, in dem man unter folgenden Menüs wählen kann:

- Spielen: Hier kann man die Anzahl der Spieler wählen, sowie die Spielart (Stroke- oder Match- Play). Danach kann man in die vorgegebene Rolle eines Spielers schlüpfen und dessen Attribute (Na-men, Schlagweite, Fähigkeiten) übernehmen oder verändern. Jetzt braucht man nur noch den Golfplatz zu wählen und spielen.

- Üben: Damit läßt sich ein Loch auf einem beliebigen Platz bespielen.

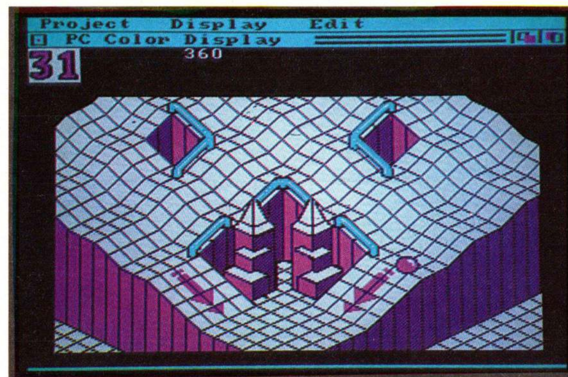
- Konstruieren: Mit diesem Menü können komplette Golfanlagen bis ins Detail angelegt, bestehende verändert oder gelöscht werden.

Golf unterstützt CGA -, EGA -, Hercules - und EGA (mono) - Karten, die mit verschiedenen Start-Parame-tern angesprochen werden. Bei der EGA - Karte lassen sich 3 Farben (grün, hellblau und braun) in ihrer Intensität steuern. Sehr empfehlenswert.

Hersteller: Electronic Arts

MARBLE MADNESS

Marble Madness, das berühmte Kugelspiel, gibt es nun endlich auch für den IBM. Spielbar ist es mit Tastatur und/oder Joystick. Die Bedienung mit der Tastatur ist ein echtes Trauerspiel, denn leider läuft die Steuerung über die berühmte Tastenkombination "QWE ASD YXC" (deutsche Tastatur), wo man im Eifer des Gefechtes öfter danebengreift. Das hätte sich besser über den Zehnerblock machen lassen. Wenn man mit 2 Spielern spielt, ist ein Farbbildschirm erforderlich, da man sonst keinen Unterschied zwischen den Kugeln sieht, ansonsten läßt sich das Spiel auch am Monoschirm problemlos spielen. Mit der Shift-Taste kann man die Kugel(n) beschleunigen, desweiteren kann man im 1-Spieler-Modus zusammen mit einem zweiten Spieler über 2



Steuereinrichtungen 1 Kugel steuern. Gute Grafik, aber nur mit Joystick zu empfehlen.

Hersteller: Electronic Arts

STREET SPORTS BASKETBALL



Dieses Spiel bietet 4 verschiedene Spielplätze an; da wären der Schulsportplatz, die Seitenstraße, eine Vorstadtstraße, sowie ein Parkplatz in der City. Unter 3 Schwierigkeitsgraden läßt sich wählen, und nachdem man seinem Team den Namen gegeben hat, wirft man mit seinem Gegner die Münze ums erste Wahlrecht. Auszuwählen aus 10 Personen, die sich selbst vorstellen, sind 3 Mitspieler. Anschließend wird der zu erreichende Endpunktestand vorgewählt, und dann geht's los. Von den 3 Mitspielern kann verständlicherweise immer nur einer gesteuert werden. Wenn ein anderer Spieler gesteuert werden soll, wird die Shifttaste niedergedrückt, bis die Kennzeichnung (gestreiftes Trikot) auf den anderen Spieler übergegangen ist. Durch schnelles Dribbeln wird der Gegner verunsichert, und man kann die gegnerischen Reihen durchbrechen. Zur Ballabgabe muß man sich dem Mitspieler einfach gegenüberstellen (Nase an Nase); das Ballabnehmen vom Gegenspieler erfolgt genauso. Gespielt werden kann alleine gegen den Computer oder zu zweit, Joystick-Unterstützung inclusive, aber auch mit der Tastatur läßt sich das Spiel noch gut bedienen.

Hersteller: Epyx

VON SVEN STILLICH, OLIVER SIEBENHAAR, RALF GÖRLACH

MIT DEM RECHNER AUF DU UND DU

Ein Assemblerkurs für Einsteiger Teil 5: Ran an's System

Wie versprochen, werden wir uns dieses Mal mit der Programmierung des Betriebssystems beschäftigen und ein erstes Programm schreiben, das einen Bildschirm und ein Fenster öffnet und in diesem einen Text ausgibt. Wir wollen jedoch (zur Ehrenrettung) darauf hinweisen, daß wir danach nicht mehr viel auf Betriebssystemroutinen eingehen werden, da diese in C geschrieben und deshalb größtenteils sehr langsam sind. Nur die systemunterstützenden Routinen der EXEC Library (siehe unten) werden vorrangig Verwendung finden. Wir wollen ja auch unseren Rechner etwas näher kennen- und begreifen lernen und nicht auf 'vorgekaute' Routinen zurückgreifen. So, das reicht höchstwahrscheinlich, um den Assembler-C-Krieg neu anzuheizen.

Nun frisch ans Werk. Die Betriebssystemroutinen sind je nach Aufgabengebiet in sogenannten Libraries (Bibliotheken) zusammengefaßt, die alle der Hauptbibliothek, der sogenannten EXEC-Library, untergeordnet sind. Einige Libraries befinden sich nach dem Booten der Kickstart (Diskette, nicht Zeitschrift) oder nach dem Einschalten (500/2000'er) bereits im Speicher, andere (zum Beispiel die 'diskfont.library') müssen erst von der Diskette nachgeladen werden. Dort müssen sie sich im Unterdirectory 'libs' befinden. Hier eine Aufstellung der AMIGALibraries:

```
   exec.library (Multitasking, I/O, Speicherreservierung, ...)
   dos.library (Diskettenverwaltung, ...)
   intuition.library (Screens, Windows, Menüs, Gadgets, ...)
   clist.library (Bearbeitung von Copperlisten)
   console.library (Textroutinen für 'CON:' Fenster)
*  diskfont.library (Verarbeitung von Zeichensätzen auf Diskette)
   graphics.library (Blittersteuerung, graphische Grundfunktionen)
*  icon.library (Bearbeitung von Workbench Symbolen)
   layers.library (Bearbeitung des Bildschirmspeichers)
   mathffp.library (mathematische Fließkommaoperationen)
*  mathieeedoubbas.library (mathematische Integeroperationen)
*  mathtrans.library (höhere mathematische Funktionen)
   potgo.library (Auswertung von analogen Inputs)
   timer.library (Programmierung für Zeitintervalle)
*  translator.library (Programmierung des Sprachsynthesizers)
*  info.library (Verarbeitung von '.info' Files)

[* = müssen von der Diskette nachgeladen werden]
```

Librarybenutzung

Will man auf die sich in den Libraries befindlichen Routinen zugreifen, müssen diese (die Libraries) erst geöffnet werden. Man erhält dadurch die Basisadresse der gewünschten Library im Speicher (Wie das genau geht, wird später genau erklärt, keine Angst!). Um nun eine Routine zu benutzen, die in dieser Bibliothek enthalten ist, wird zu der Basisadresse ein negativer Offset addiert. Das Ergebnis ist die Adresse, die angesprungen werden muß, um die Routine auszuführen. Davor müssen bestimmte Register mit Werten geladen werden, die als Variablen der Betriebssystemroutine zur Bearbeitung übergeben werden. Hat die Routine ihre Aufgabe erfüllt und springt ins eigentliche Programm zurück, übergibt sie diesem im Datenregister 0 (D0) meistens das Ergebnis ihrer 'Bemühungen'. Ist während der Abarbeitung der Betriebssystemroutine ein Fehler aufgetreten, so wird in D0 der Wert Null übergeben. Diese Null kann man dann vom Programm aus abfragen. Aus Platzmangel ist es uns nicht möglich, sämtliche Offsets aufzulisten. Wir werden nur diese angeben, die für unser Programm notwendig sind. Vielleicht wird in einer späteren KICKSTART-Ausgabe eine komplette Liste abgedruckt. Die Offsets sind jedoch vielen gängigen Publikationen (Büchern, ...) zu entnehmen. Nun ans Werk !

Wir öffnen unsere erste Library

Die ersten Programmzeilen werden immer den Deklarationen von Offsets, Konstanten usw. gewidmet (siehe Zeilen bis 15). Dann kann unser Programm auch schon beginnen. Als erstes werden wir die 'intuition.library' öffnen, da diese für das Erstellen von Screens und Windows zuständig ist. Es ist ein ungeschriebenes Gesetz unter uns Assemblerprogrammierern, daß sich die Basisadresse der aktuell benutzten Library in A6 befinden muß. Und das ist auch schon unser erster Schritt:

```
17 move.l  Execbase,a6
```

Damit ist die EXEC Library als aktuelle Library definiert. Ihre Basisadresse befindet sich immer in $04 (Execbase = $04). Nun benötigt die Oldopenlibrary-Routine in A1 die Startadresse einer mit einer Null abgeschlossenen Zeichenkette, den Namen der jeweiligen Library. Diese Parameter (Zeichenketten, Variablen, ...) werden GRUNDSÄTZLICH am Ende eines Programmlistings definiert. Dies

ist damit zu begründen, daß alle Zeichen ja als Zahlenfolgen im Speicher abgelegt sind. Auch jeder Maschinensprachebefehl entspricht einer Zahlenfolge. Trifft der Prozessor aber auf eine Zeichenkette (zum Beispiel 'intuition.library'), so ergeben diese Werte unsinnige oder gar keine Befehle. Dies verwirrt den Prozessor so sehr, daß er sich entnervt in die ewigen Bytegründe begibt und nur durch emsiges CTRL/AMIGA/AMIGA-Drücken wiederzubeleben ist (das Definieren der Zeichenkette geschieht übrigens in Zeile 78).

Dann wird der Routine diese oben erwähnte Startadresse übergeben. Sie ist ebenfalls in Zeile 78 definiert ('INTname'):

```
18 lea      INTname,a1
19 jsr      Oldopenlibrary(a6)
```

Nun wird die EXEC Routine 'OldOpenLibrary' angesprungen. Diese Routine 'öffnet' die Intuition Library und gibt deren Basisadresse im Speicher in D0 zurück. Ist der Inhalt des Datenregisters Null, wird dies abgefragt und ans Ende des Programmes gesprungen. Ansonsten wird die Basisadresse in die Variable 'INTbase' gerettet:

```
20 beq      Ende
21 move.l   d0,INTbase
22 move.l   d0,a6
```

In Zeile 18 wird die Intuition Library als aktuelle Library ernannt - ihre Startadresse wird in A6 geschoben. Wir befinden uns nun in der Intuition Library (Huch!).

My Screen is my Castle

Wollen wir nun einen eigenen Bildschirm öffnen, so braucht die dafür benutzte Routine ('OpenScreen') für ihren Aufruf einen Zeiger auf eine Struktur. Eine Struktur ist eine Folge von Zahlenwerten, deren Datenlänge (Byte, Word, ...) fest vorgeschrieben ist, da die Routine diese Struktur in einer gewissen Weise interpretiert und verarbeitet. Ein Beispiel ist die Screen-Struktur, die wir hier brauchen. Diese ist im Listing in Zeile 89 definiert. Hierzu nur die Syntax der Struktur:

```
dc.w    LeftEdge,TopEdge,Width,Height,
        Depth
dc.b    DetailPen,Blockpen
dc.w    ViewModes,Type
dc.l    TextAttr,DefaultTitle,Gadgets,
        Bitmap
```

Gewissenhafte Leser unserer Zeitschrift wissen bereits, um was es sich bei diesen Werten handelt (KICKSTART Juni '87/Intuition Kurs Seite 13). Da in diesem Kursteil das Öffnen eines Fensters nur mittels C erklärt wurde (wobei man in dieser Sprache zum Beispiel die Viewmodes durch Einsetzen der jeweiligen Schlüsselwörter [INTERLACE|SPRITES|...] setzen kann, bei Assembler jedoch die Zahlenwerte angeben muß), sollen in diesem Kurs die Zahlenwerte der am meisten benutzten Grundmodes von Fenstern und Bildschirmen noch einmal aufgeführt werden:

```
INTERLACE          $0004
HOLD & MODIFY      $0800
EXTR AHALFBRITE    $0080
SPRITES            $4000
DUALPLAYFIELD      $0400
HIRES              $8000
```

Natürlich können diese Viewmodes untereinander auch kombiniert werden. Sollen sich zum Beispiel Sprites auf einem Hiresscreen tummeln, wäre für den Viewmode $C000($4000+ $8000) einzusetzen. Es ist zu beachten, daß es sich hierbei um einen Wortwert handeln muß (siehe Screen-Struktur). Doch nun weiter in unserem Projekt. Es soll ein Bildschirm mit der Struktur (Startadresse als 'OSargs' definiert), die in Zeile 89 festgelegt ist, geöffnet werden:

```
24  lea      OSargs,a0
25  jsr      OpenScreen(a6)
26  beq
             Ende
27  move.l   d0,Screen
```

In den Zeilen 24 und 25 wurde also ein Bildschirm geöffnet. Ist dabei ein Fehler aufgetreten, erfolgt ein Sprung zu 'Ende' (Zeile 64). Ansonsten wird der von der Routine erhaltene Pointer in 'Screen' geschoben. Die Erklärung dieser Zeile befindet sich unten. Sie sind nun also der glücklicher Besitzer eines eigenen Bildschirms (Abspeichern nicht vergessen !!!).

Das Fenster zur Welt

Wenn bis jetzt alles klar ist, wagen wir uns ans Öffnen eines Windows. Wie beim Screen brauchen wir hierfür eine eigene Struktur (Sie merken schon, man braucht für fast alles eine Struktur). Und hier ist sie auch schon:

```
dc.w LeftEdge,TopEdge,Width,Height
dc.b Detailpen,BlockPen
dc.  IDCMPFlags,Flags,Gadget,Image,
     Title
dc.l Screen,Bitmap
dc.w MinWith,MinHeight,MaxWidth,
     MaxHeight
dc.w Type
```

Wenn Sie sich die Windowstruktur so ansehen, fällt Ihnen bestimmt der Parameter 'Screen' auf. Na? Richtig, davon haben Sie doch schon einmal gehört. Wenn Sie im Listing nachsehen, finden Sie genau vor diesem Wert das Label 'Screen'. Und sicherlich werden Sie daraus auch richtig schlußfolgern, daß hier dem Window der Pointer zu seinem Screen übergeben wird. Auf diese Weise erfolgt die Vernetzung beiden Strukturen (sonst wüßte das Fenster ja gar nicht, wo es hingehört und würde hilflos und allein im Speicher herumirren). Die Windowstruktur unseres Programmes finden Sie ab Zeile 95 im Listing. Natürlich wurde in der KICKSTART - wie es sich gehört - auch diese Struktur schon näher erläutert (Ausgabe Juli/August 1987/ Intuition Kurs-Seite 17ff). Hierzu (alles auf einmal, puh...) dann auch noch die grundlegenden Flags (NICHT IDCMP FLAGS, das ist etwas anderes):

```
WINDOWSIZIN             $0001
WINDOWDRAG              $0002
WINDOWDEPT              $0004
WINDOWCLOSE             $0008
BACKDROP                $0100
BORDERLESS              $0800
ACTIVATE                $1000
```

Auch hierbei muß es sich diesmal um Langwortwerte handeln. Nun übergeben wir unserer Routine OpenWindow noch die Struktur (im Listing in Zeile 95 als 'OWargs' definiert) und schon hat unser Bildschirm ein Fenster...

```
29 lea      OWargs,a0
30 jsr      OpenWindow(a6)
31 beq      Ende
32 move.l   d0,a0
33 move.l   d0,Window
```

Sehen Sie, war doch überhaupt nicht so schwer. Und, haben wir in der letzten Ausgabe zu viel versprochen ? Hier haben sie ein Programm, mit dem sie nach Herzenslust herumexpermentieren können. Aber es kommt noch besser ...

Was man schwarz auf weiß besitzt ...

Wir wollen nun in einer geistigen Höchstleistung versuchen, in unserem Fenster den Namen eines von einer bekannten Publikation abstammenden Betriebsystems unterzubringen (Sie ahnen es schon - KICKSTART !?!). Dazu brauchen wir aber den RastPort. Er ist eine Art Zusatzstruktur, der für jedes Fenster spezifisch ist und das jeweilige Aussehen angibt. Ein Zeiger auf den RastPort liegt in jeder Fensterstruktur (die Struktur, die man nach dem Öffnen erhält (D0), nicht die eigene Struktur) ab dem 50sten Byte. Somit ist auch der Befehl in Zeile 34 klar:

```
34 move.l    50(A0),RastPort
```

(Zur Anfangsadresse der Window-Struktur, die sich in A0 befindet, wird dez. 50 hinzuaddiert. Der Inhalt dieser Adresse ist der Zeiger auf den Rastport). Weiter öffnen wir im Programm die 'graphics.library' und erklären sie in Zeile 41 zur aktuellen Library. Nun beginnt die Schleife für die Textausgabe. Dafür benötigt man zwei Betriebssystemroutinen: Move und Text. Wie man aus dem Namen schon ersehen kann, dient 'Move' zur Bewegung des imaginären Cursors, d.h. zur Einstellung der Koordinaten, ab denen der Text gePRINTet werden soll. Diese Koordinaten werden in den Datenregistern D0 (x) und D1 (y) übergeben. 'Text' - wer hätte es geahnt ? - ist für die Textausgabe zuständig. Diese Routine benötigt als Parameter in A1 den Zeiger auf die RastPort-Struktur, in A0 die Startadresse des mit einer Null abgeschlossenen Strings (in unserem Programm in Zeile 82 definiert) und in D0 die Länge der Zeichenkette. Nun zur Programmierung der Schleife, die den Text versetzt auf den Monitor (oder Fernseher) bringen soll:

```
43 move.l #08,d5
44 move.l #50,d6
45 move.l #50,d7
```

In diesen Zeilen bereiten sich verschiedene Datenregister auf ihren Einsatz vor, d.h., man bringt sie auf den richtigen Anfangswert. In D5 befindet sich der Schleifenzähler, der bestimmt, wie oft der Text in das Fenster geschrieben werden soll. In D6 und D7 befinden sich die Koordinaten für die Move Routine.

```
47 Print:
48 move.l  RastPort,a1
49 move.l  d6,d0
50 move.l  d7,d1
51 jsr     Move(a6)
52 move.l  RastPort,a1
53 lea     String,a0
54 move.l  #13,d0
55 jsr     Text(a6)
```

Dies ist das Herz der Schleife, hier erfolgt die Positionierung und Ausgabe des Textes.

```
 1: ; ———————————————————————
 2: ; Assemblerkurs Teil V (OpenScreen/Window, Move, Text)
 3: ; by Sven Stillich, Oliver Siebenhaar
 4: ; (c) 1988 KICKSTART Ausgabe März 1988
 5: ; ———————————————————————
 6: ;
 7: OldOpenLibrary  = -408
 8: OpenScreen      = -198
 9: OpenWindow      = -204
10: Move            = -240
11: Text            = -60
12: CloseWindow     = -72
13: CloseScreen     = -66
14: CloseLibrary    = -414
15: ExecBase        = $04
16: ;
17: move.l          ExecBase,a6
18: lea             INTname,a1
19: jsr             OldOpenLibrary(a6)
20: beq             Ende
21: move.l          d0,INTbase
22: move.l          d0,a6
23:
24: lea             OSargs,a0
25: jsr             OpenScreen(a6)
26: beq             Ende
27: move.l          d0,Screen
28:
29: lea             OWargs,a0
30: jsr             OpenWindow(a6)
31: beq             Ende
32: move.l          d0,a0
33: move.l          d0,Window
34: move.l          50(a0),RastPort
35:
36: move.l          ExecBase,a6
37: lea             GFXname,a1
38: jsr             OldOpenLibrary(a6)
39: beq             Ende
40: move.l          d0,GFXbase
41: move.l          d0,a6
42:
43: move.l          #08,d5                       ; Schleifenwert
44: move.l          #50,d6                       ; x Koordinate
45: move.l          #50,d7                       ; y Koordinate
46:
47: Print:
48: move.l          rastport,a1
49: move.l          d6,d0
50: move.l          d7,d1
51: jsr             Move(a6)
52: move.l          Rastport,a1
53: lea             string,a0
54: move.l          #13,d0
55: jsr             Text(a6)
56: add.l           #08,d6
57: add.l           #10,d7
58: dbra            d5,Print
59:
60: Test:
61: cmp.b           #$39,$bfec01                 ; Auf CTRL prüfen
```

```
 62:  bne              Test
 63:  ;
 64:  Ende:
 65:  move.l           INTbase,a6
 66:  move.l           Window,a0
 67:  jsr              CloseWindow(a6)
 68:  move.l           Screen,a0
 69:  jsr              CloseScreen(a6)
 70:  move.l           ExecBase,a6
 71:  move.l           INTbase,a1
 72:  jsr              CloseLibrary(a6)
 73:  move.l           GFXbase,a1
 74:  jsr              CloseLibrary(a6)
 75:  rts
 76:  ;
 77:  even
 78:  INTname:         dc "intuition.library",0
 79:  even
 80:  GFXname:         dc "graphics.library",0
 81:  even
 82:  String:          dc.b "KICKSTART !!!",0
 83:  even
 84:  INTBase:         dc.l $0000
 85:  GFXBase:         dc.l $0000
 86:  Window:          dc.l 0
 87:  RastPort:        dc.l 0
 88:
 89:  OSargs:
 90:  dc.w 0,0,320,250,1      ; Koordinaten: 0/0-320/200 1 Bitplane
 91:  dc.b 0,1                         ; DetailPen, BlockPen
 92:  dc.w 0,15                        ; ViewMode HIRES, Customscreen
 93:  dc.l $0000,$0000,$0000,$0000      ; Kein Text, Gadget, Bitmap
 94:
 95:  OWargs:
 96:  dc.w 30,40,260,160               ; Koordinaten: 30/40-260/160
 97:  dc.b 0,1                                 ; DetailPen, BlockPen
 98:  dc.l $0000,$1007,$0000,$0000,$0000  ; ACTIVATE|WSIZING|WDRAG
 99:  Screen:                                  ; Pointer zum Screen
100:  dc.l $0000,$0000                         ; Nichts gesetzt
101:  dc.w 50,50,100,100,15                    ; Minimal x/y, ...
```

```
56 add.l  #08,d6
57 add.l  #10,d7
58 dbra   d5,Print
```

Wie Sie hoffentlich aus der Sequenz ersehen können, wird die x-Koordinate um dez. 8 erhöht, die y-Koordinate hingegen um dez. 10. Der Befehl in Zeile 58 prüft, ob d5 schon den Wert -1 erreicht hat (= Ende der Schleife). Ist dies nicht der Fall, wird zu 'Print' (Zeile 47) gesprungen. Nach Beendigung der Schleife geht das Programm in eine Warteschleife. Hierbei vergleichen wir den Inhalt des Hardwareregisters $BFEC01, in welchem die gedrückten Sondertasten angezeigt werden, mit dem Wert $39 (für die Control-Taste). Stimmt der Wert mit dem Inhalt des Registers überein, wurde die angegebene Sondertaste gedrückt, und unser Programm fährt fort (wohin ?) und findet ein stilvolles Ende.

Man kann natürlich auch andere Sondertasten außer der CTRL-Taste abfragen, als da wären:

$3F	SHIFT links
$3D	SHIFT rechts
$37	ALTERNATE
$33	AMIG Alinks
$31	AMIG Arechts

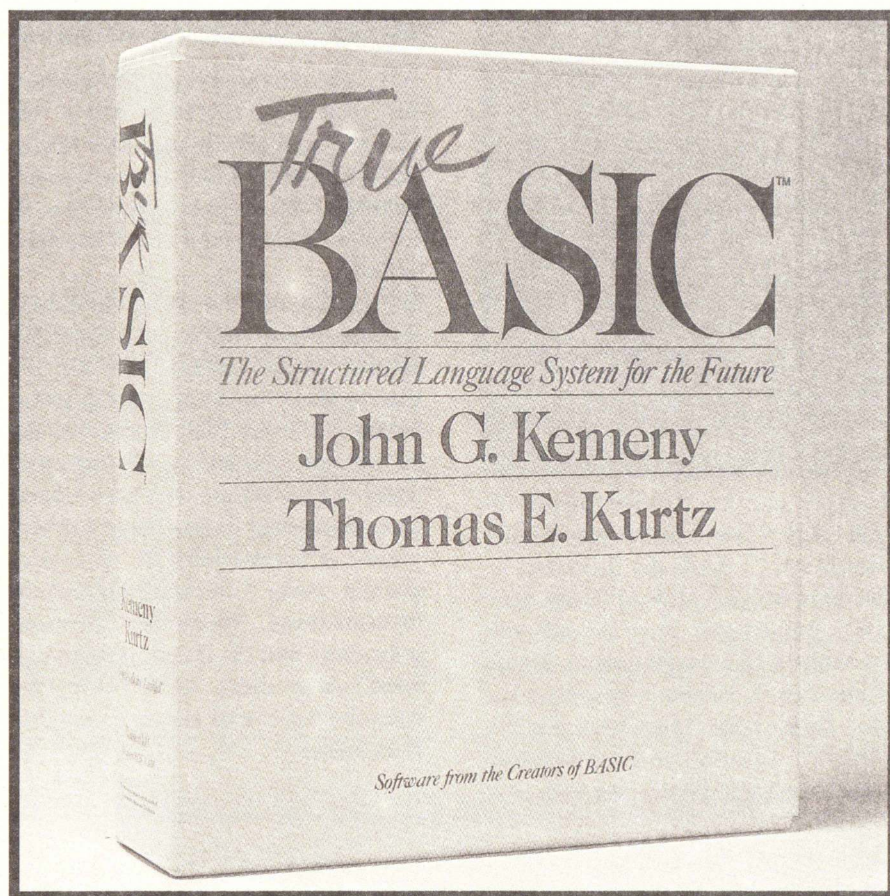
Am Ende unseres Programmes schließen wir alle Libraries sowie das Window und den Screen. Das Schließen des Windows muß immer vor dem Schließen des Screens geschehen, da sich das Fenster ja auf dem Bildschirm befindet.

Wir hoffen, daß wir Ihnen mit diesem Kursteil einen (wenn auch nicht besonders tiefen) Einblick in die Betriebssystemprogrammierung gegeben haben. Nächstes Mal führen wir das Projekt fort, d.h wir installieren einen anderen Zeichensatz, zeichnen Linien, usw. Außerdem (wenn unser Rechengenie Ralf wieder dabei ist) verwenden wir ein wenig Zeit und Platz auf die verschiedenen möglichen Bitmanipulationen und die dazugehörigen Befehle. Also dann, tschüß bis zum nächsten Teil - und immer ein Bit übrigbehalten ...

VON CHRISTIAN KELLER

TRUE FOR BLUE

TrueBasic auf MSDOS



Auf dem Sektor der Programmiersprachen geht es mittlerweile zu wie in Babylon: Immer besser, aber keiner versteht den anderen. Das fängt schon bei der Sprache für einen Rechner an und wird gänzlich verwirrend, wenn man auf einen anderen Rechner wechselt. Gute Ansätze zur Besserung zeigen sich nun aber in den Software-Schmieden, so zum Beispiel der ISO-Standard bei Pascal.

Das Softwarehaus Pfotenhauer beschreitet hingegen einen anderen Weg zur Standardisierung. Man hat sich dazu entschlossen, einen Basic-Dialekt auf den Markt zu bringen, der zwar zu nichts kompatibel, dafür aber auf vielen handelsüblichen Rechnern implementiert ist. Hierzu zählen zur Zeit AMIGA, Atari ST, Apple MacIntosh, IBM-PC und Konsorten. Die Amiga-Version wurde bereits von uns getestet. Hier soll jetzt einmal die Ausgabe für den IBM-PC unter die Lupe genommen werden.

TrueBasic stammt von John G. Kemeny und Thomas E. Kurtz, den Vätern von Basic. Das läßt ja schon einiges an Spannung bezüglich dieser Sprachversion entstehen. Die Grundversion von TrueBasic umfaßt ein Benutzerhandbuch, ein Referenzhandbuch und zwei Programmdisketten. Die Handbücher, zwei etwa DIN A5-große Ringbücher, machen einen guten Eindruck. Sie sind flüssig geschrieben und mit vielen Grafiken erweitert.

Das Ruder in der Hand

Nach dem Aufrufen erscheint ein zweigeteilter Bildschirm: Die oberen zwei Drittel werden vom Edierfenster in Anspruch genommen, das untere Drittel dient als Kommandofenster und zur Ausgabe von Meldungen.

Dazwischen findet man eine Leiste, auf der die Belegung der Funktionstasten angezeigt wird. Bei TrueBasic gibt es (leider) keine Menüleisten, wie sie sich bei den Programmiersprachen langsam aber sicher durchsetzen. Das verkompliziert die Arbeit ein klein

Benchmark	Truebasic	TurboBasic sng	TurboBasic int	GWBasic sng	GWBasic int
1	1.3	30.4	0.1	11.0	7.8
2	2.8	46.1	0.3	31.8	27.4
3	5.5	48.5	0.9	33.5	30.9
4	110.2	55.7	27.4	35.8	34.1
5	41.9		2.6	>1000	

wenig. Eine Online-Hilfe ist implementiert. Den Befehl "Load" sucht man vergebens, er heißt hier "old".

Was etwas seltsam anmutet und Truebasic wenig professionell erscheinen läßt, ist die Art und Weise, wie es aufgerufen und verlassen wird. Mit "hello" fängt man an und mit "Bye" verläßt man es. Speziell das Beenden von TrueBasic fällt ohne Handbuchstudium schwer (Ich habe 5 Minuten herumprobiert: End, Exit, System, Quit usw.). Hier sollte doch Zweckmäßigkeit und Einheitlichkeit vor Originalität gehen !

Compiler oder Interpreter ?

Diese Frage ist auf Anhieb nicht ohne Zweifel zu beantworten. Die Programme sind nicht alleine lauffähig, was ein typisches Interpreter-Merkmal ist. Trotzdem ist TrueBasic ein Compiler. Der Haken an der Sache: Es handelt sich um einen Compiler, der in der Standardversion nicht fähig ist, die Library-Module an die Programme anzulinken. Deshalb müssen diese von der Entwicklungsumgebung zur Verfügung gestellt werden. Wenn oben von Standardversion geredet wird, kann man sich denken, daß es auch eine Ausführung gibt, die zum Erzeugen von lauffähigen Stand-Alone-Programmen fähig ist. Dieses Zusatzpaket nennt sich TrueBasic Runtime-Paket und enthält den Linker und einige Zusatzprogramme. Damit ist TrueBasic dann in der Lage, .EXE-Dateien zu erzeugen.

Einen großen Vorteil hat die Verfahrensweise mit getrenntem Compiler und Binder bei TrueBasic: Kompilierte Programme sind direkt übertragbar. Man kann also ein Programm auf dem IBM entwickeln, kompilieren, auf den AMIGA übertragen, starten und es läuft.

Strukturiertes Programmieren

TrueBasic bietet viele Befehle an, die ein strukturiertes und übersichtliches Programmieren ermöglichen. Neben der einfachen FOR...NEXT-Schleife gibt es jetzt auch die Anweisungen DO WHILE... LOOP und DO UNTIL...LOOP. Die IF..THEN-Anweisung wurde um die Befehle ELSE und ELSEIF erweitert. Gänzlich neu in Basic ist die Pascal-ähnliche Funktion:

```
SELECT CASE Ausdruck
  CASE ....
  CASE ....
END SELECT
```

Unterprogramme, Funktionen und Libraries werden auch unterstützt. So ist es nun auch in Basic möglich, große Probleme in kleine Teillösungen aufzusplitten und so die Überschaubarkeit erheblich zu verbessern.

Von Bildern und Gemälden

Ein Highlight unter den Fähigkeiten von TrueBasic ist die Grafik. Die Sprachdefinition ist in diesem Bereich besonders reichhaltig ausgefallen. Das fängt schon bei den unterstützten Grafikmodi an. CGA-Medres, CGA-Hires, Hercules und EGA-Grafik werden bewältigt. Neben den Standard-Grafikbefehlen werden eine ganze Reihe von zusätzlichen Anweisung angeboten, die vorallem das Programmieren von bewegten Grafiken erleichtern. Besonders erwähnenswert erscheint mir hier der "Set Window"-Befehl. Dieser Befehl ermöglicht es, eine eigene Skalierung an den Bildschirm anzubringen. Gibt man zum Beispiel "Set Window -pi,pi,-1,1" ein, hat der rechte obere Bildpunkt die Koordinaten -pi,1 und der linke untere pi,-1. Die Umrechnung von Benutzerkoordinaten in "physikalische" Bildschirmpunkte übernimmt TrueBasic ganz. So hat man es sehr einfach, schöne Grafiken auf allen Bildschirmadaptern mit einer einfachen Programmlösung zu erstellen. Ebenfalls ausgefallen sind die Befehle zum Zeichnen aus Matrizen heraus. Dazu dienen die Befehle "MAT PLOT POINTS, MAT PLOT LINES und MAT PLOT AREA". Das Zeichnen von Drahtmodellen und deren Bewegung ist so kein Problem mehr.

Mathe: 1

Wer mathematische Probleme zu lösen hat, wird von TrueBasic angenehm

```
mathematische Funktionen
  ABS      DIVIDE EPS     EXP      FP
  INT      IP     LOG     LOG2     LOG10
  MAX      MAXNUM MIN     MOD      REMAINDER
  RND      ROUND  SGN     SQR      TRUNCATE
trigonometrische Funktionen
  ANGLE    ATN    COS     DEG      PI
  RAD      SIN    TAN
Matrix-Operationen
  MAT READ    MAT INPUT     MAT LINE INPUT
  MAT REDIM   MAT PRINT     Matrixaddition
  Matrixsub.  Matrixmult.   Skalarmult.
  Matrixinvert.  Transposition
  CON      IDN    NUL$    ZER      DET
  DOT
```

Die mathematischen Funktionen von True Basic

überrascht sein. Die Implementierung von mathematischen Funktionen ist ausgesprochen reichhaltig ausgefallen und bietet neben den Standardfunktionen noch einige Besonderheiten, die man nur selten oder garnicht in eine Sprache eingebunden findet. Bei den normalen Funktionen wäre zum Beispiel der Befehl DIVIDE zu nennen, der eine Division ausführt, und als Ergebnis den ganzzahligen Teil des Quotienten und den Rest liefert. REMAINDER liefert nur den Rest einer Division. Mit TRUNCATE ist es möglich, eine Zahl auf eine bestimmte Anzahl von Dezimalstellen zu kürzen. An trigonometrischen Funktionen sind die Umrechnung von Bogenmaß in Grad hinzugefügt worden und die Berechnung des Winkels zwischen einem Punkt und der x-Achse.

Ganz besonders hat mir die Implementierung von Matrizenoperationen gefallen. Matrizen haben heutzutage eine große Bedeutung in der Mathematik und der Ingenieurwissenschaft und keiner, der sich mit der Lösung von technischen Problemen befaßt, wird an der linearen Algebra vorbeikommen. Die Bedeutung dieses Zweiges der Mathematik wird in keiner anderen Sprache richtig gewürdigt. TrueBasic bietet hier aber einiges. Angefangen bei der einfachen Matrizen-Addition und -Subtraktion über die Multiplikation Skalar-Matrix oder Matrix mal Matrix bis hin zur Invertierung sind viele Möglichkeiten gegeben. Mit den oben schon erwähnten Matrix-Grafikbefehlen zusammen ist dem Programmierer ein mächtiges Werkzeug in die Hand gelegt. Ebenso werden einige Standardmatrizen wie die Einheitsmatrix oder die Null-Matrix angeboten. Insgesamt lassen die Matrizen-Operationen TrueBasic in einem guten Licht dastehen. Mathematiker und Ingenieure haben mit TrueBasic auf jeden Fall einen guten Griff gemacht.

TrueBasic GTI

Was natürlich nicht fehlen darf, sind die obligatorischen Zeitmessungen. Der direkte Vergleich mit anderen Basic-Dialekten fällt aber schwer, da TrueBasic nur einen Zahlentyp kennt. Andere Sprachdefinitionen kennen die Typen Integer, Real und Doublereal. TrueBasic berechnet hingegen immer auf 10 Stellen genau. Unter diesem Gesichtspunkt betrachtet steht TrueBasic bei den Zeitmessungen sehr gut da. Man kann ihm ohne Zweifel die Trophäe für schnelle Real-Arithmetik zuerkennen. Nur bei Anwendungen, die überwiegend Integer-Zahlen benutzen, kann TurboBasic auftrumpfen. Insgesamt sollte man den Vergleich mit Vorsicht genießen. Aber eins steht fest: TrueBasic ist auf jeden Fall ausreichend schnell.

TrueBasic GLX

"GLX" wie "Großer Luxus". Wie bei Automobilen gibt es einiges Zubehör gleich dazu, anderes muß man zusätzlich erstehen. Mitgeliefert werden mehrere Programmbibliotheken. Es sind deren acht. Vier davon bieten mathematische Funktionen, drei enthalten Unterprogramme zur Grafikerzeugung und eine dient zur Menüerzeugung. Die mathematischen Libraries liefern diverse hyperbolische Funktionen, zusätzliche trigonometrische Funktionen in Grad und Bogenmaß und eine Reihe mathematischer Operationen wie Fakultät, Binomialkoeffizient und -verteilung, Poisson-Verteilung oder Logarithmus von x zur Basis b. Die Grafikbibliotheken sind ebenfalls reichhaltig und sinnvoll bestückt. Es gibt Befehle, um automatisch Balkendiagramme aus DATA-Zeilen zu bilden oder um Funktionen zu zeichnen. Außerdem gibt es Funktionen zum Zeichnen und Füllen von Kreisabschnitten und Ellipsen. Die Menü-Library schließlich ermöglicht es, Menüleisten zu erzeugen, die über die Funktionstasten angewählt werden. Wem das alles nicht genügt, kann sich noch zusätzliche Bibliotheken kaufen oder andere Erweiterungen erstehen. Als Library wäre die Such- und Sortier-Bibliothek zu nennen. Um Programme aus anderen Basic-Dialekten zu übersetzen, gibt es den PC-Converter. Dieser nimmt die Umsetzung von BasicA-Programmen, soweit es geht, automatisch vor. Letztendlich sei noch das Runtime-Paket erwähnt, das die Erzeugung von .EXE-Dateien ermöglicht.

Das einzig Wahre ?

TrueBasic hinterläßt einen guten Eindruck. Von der Bedienung her etwas unkomfortabel, kann es im grafischen und mathematischen Bereich aber voll überzeugen. Speziell Naturwissenschaftler und Ingenieure werden die mathematischen Fähigkeiten zu schätzen wissen. Ebenso werden Programmierer von Anwendungen, die grafische Elemente benötigen (Business-Grafik, Funktionen, Graphen usw.), ihre Freude an TrueBasic haben. Heutzutage gibt es wohl keine schlechten Programmiersprachen mehr. Das Augenmerk sollte sich daher auf besondere Stärken richten. Und die hat TrueBasic: Grafik und Mathematik.

+ portabler Kode
+ sehr viele mathematische Funktionen
+ sehr gute Grafikfähigkeiten
+ Erweiterungsmöglichkeiten
+ Strukturbefehle
- in Grundausführung keine Stand-alone-Programme
- Editor etwas gewöhnungsbedürftig

```
PC Monochrone Display
OPTION angle degrees

SET MODE "egahires"
OPEN #1: screen .22,.78,.1,.9
SET window -1,1,-1,1

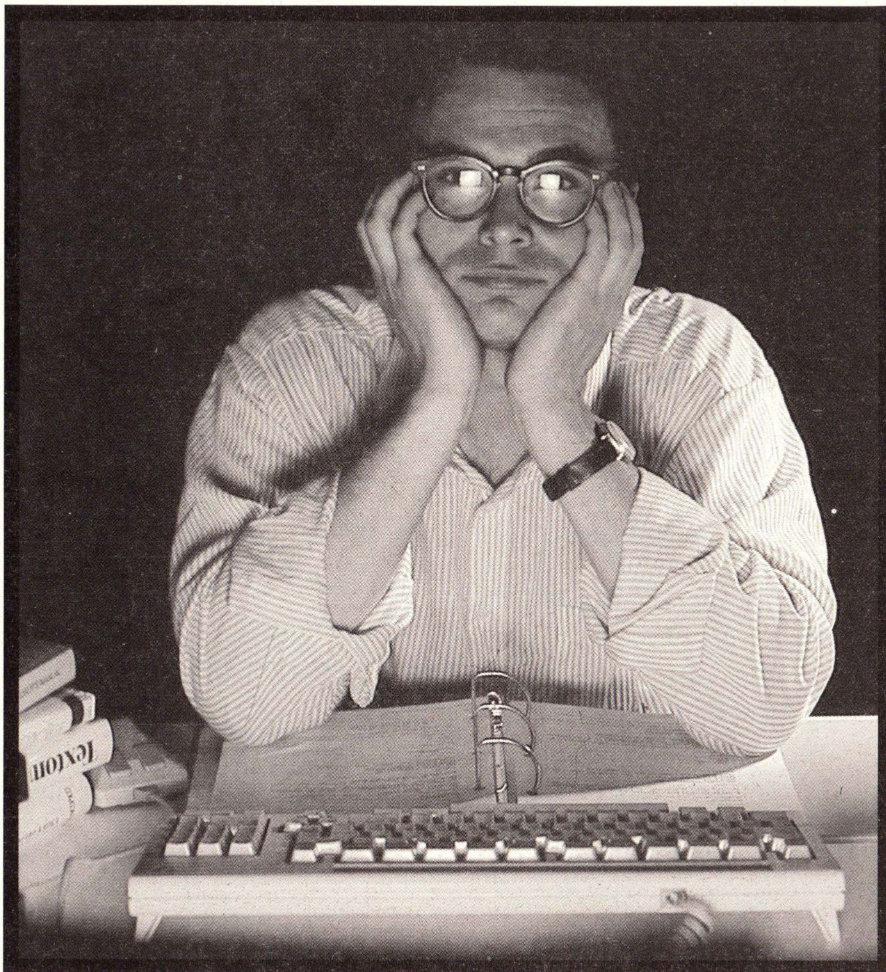
CALL colors               ! All colors
CALL mix                  ! Initial mix
FOR c = 1 to 12           ! 12 sectors of circle
    SET COLOR c
    LET a = (c-1)*30
    LET a2 = c*30
    PLOT AREA: 0,0; cos(a), sin(a); cos(a2),sin(a2)
    SET COLOR 15        ! Outline in black
    PLOT cos(a), sin(a); cos(a2),sin(a2)
NEXT c

wheel.tru
F1-EDIT F2-CMD F3-FIND F4-MARK F5-COPY F6-MOVE F7-UNDEL F8-BRKPT F9-RUN F10-HELP
WINDOWS.TRU     WURZEL.TRU      XREF.TRU        ZINSEN.TRU
Ok. old autos.tru
Ok. old staaten.tru
Ok. old test.tru
Ok. old wheel.tru
Ok.
```

VON THOMAS TROST

VOM PROBLEM

zum Programm



Die Kenntnis einer Programmiersprache genügt nach Ansicht vieler Computerbenutzer zum Erstellen eines Programmes. Wer sich aber voller Elan hinter die Tastatur klemmt, einen Editor aufruft und drauflos schreibt, wird bald eines Besseren belehrt.

Es ist natürlich durchaus möglich, ein Programm, welches die Summe dreier Zahlen ermittelt, aus dem Kopf heraus zum Laufen zu bringen, bei der Erstellung komplexerer oder mathematisch schwieriger Programme bedarf es jedoch einer nicht unerheblichen Vorbereitung. Der folgende Artikel versucht nun, Ihnen eine Vorgehensweise nahezulegen, die es Ihnen erleichtern soll, ein Programm zur Lösung eines Problems zu erstellen.

Je nach Umfang des Problems gibt es verschiedene Vorgehensweisen bei der Programmerstellung. Man kann sich aus einer Programmiersprache Anweisungen aussuchen und zusammensetzen oder das Problem von Hand rechnen und sich die passenden Befehle herauspicken. Dies eignet sich jedoch nur bei sehr kleinen, einfachen Programmen.

Abhilfe schaffen hier formalisierte Methoden wie Softwareengineering oder strukturierte Programmierung (Top-Down- oder Bottom-Up Methode). Für den Anwender zu Hause, der allein eine komplexe Aufgabe bewältigen will, eignet sich die hier vorgestellte Entwurfstrategie. Grob gegliedert sieht sie folgendermaßen aus:

— Analyse des Problems
— Erstellung eines Algorithmus
— Entwurf eines Programms
— Codierung in eine Programmiersprache.

Ein Programm ist definiert als Ablauf- oder Arbeitsplan zur Lösung eines Problems. Um einen solchen Plan zu realisieren, bedient man sich

einiger Hilfsmittel. Die zunächst naheliegensten sind Papier, Bleistift und einige "Minuten" Bedenkzeit, in der Sie sich klar werden sollten, was Sie eigentlich machen wollen. Das Problem muß formuliert werden, d.h. Sie sollten sich Gedan-ken über Lösungsweg, Rechenvorschriften und Algorithmus machen. Stellen Sie sich z.B. folgende Fragen:

- — Was ist gegeben?
- — Was ist bekannt?
- — Was ist gesucht?
- — Wie kann man das Problem darstellen?
- — Ist die Lösung bekannt?
- — Ist das Problem lösbar?

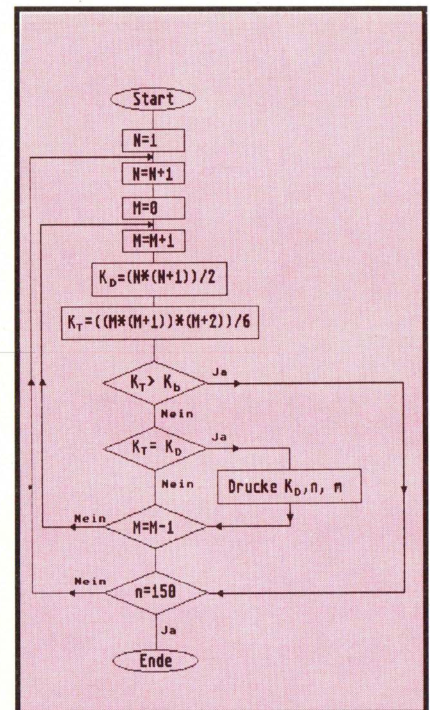
Ein Problem läßt sich auf verschiedene Arten darstellen und anhand der gewählten Darstellung konkretisieren (z.B. in Text, Graphik, Formeln oder Tabellen). Machen Sie sich das an dem Beispiel am Ende des Artikels bewußt. Es ist wichtig, daß Sie die Antworten auf Ihre Fragen in irgendeiner Art, sei es durch Text oder Skizzen, dokumentiert haben und Ihr Problem exakt umrissen haben. Damit ist der erste Schritt auf dem Weg vom Problem zum Programm getan. Sie können nun dazu übergehen, aus Ihren Aufzeichnungen heraus einen (Lösungs-)Algorithmus zu entwickeln. Unter einem Algorithmus versteht man eine Reihenfolge von Anweisungen, mit deren Hilfe ein gegebenes Problem gelöst werden kann.

Man unterscheidet zwischen linearen, verzweigten, iterativen und rekursiven Formen von Algorithmen. Linear nennt man eine einfache, sequentielle Folge von Anweisungen mit einem festen, vorgegebenen Weg, im Gegensatz zum verzweigten Algorithmus, der die Struktur eines Entscheidungsbaumes aufweist. Iteration ist eine mehrfach gesetzte Folge (Schleife), und die Zurückführung auf einen einfachen, bekannten Fall nennt man Rekursion.

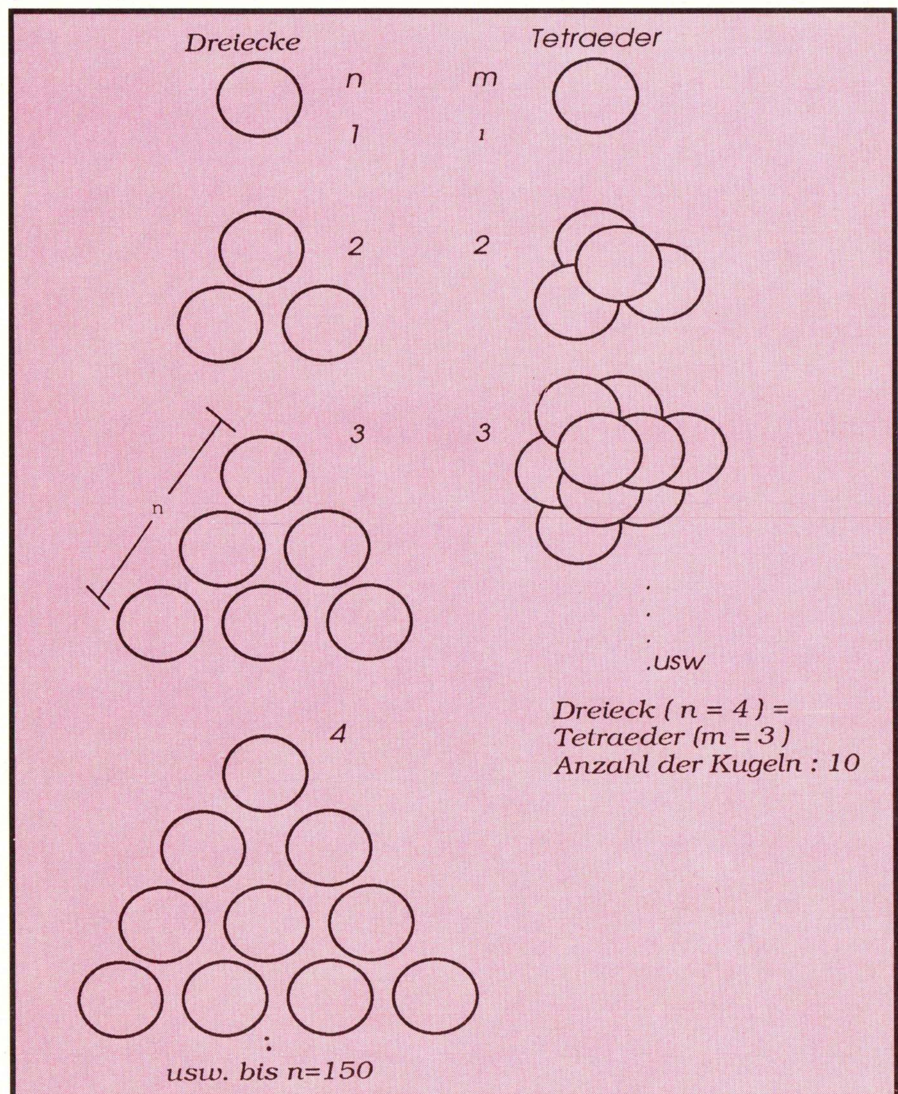
Da für eine Vielzahl von Problemen schon Algorithmen entwickelt worden sind, lassen sich diese bei Ihren Anwendungen wiederverwenden, eventuell in einer dem Problem angepaßten Form. Es wäre z.B. bei einem Sortier-Problem nicht angebracht, einen neuen Algorithmus zu entwickeln, da hierfür schon eine Reihe existieren (siehe: KICKSTART 9/'87; Sortier-Probleme). Unbekannte Algorithmen sollten Sie auf bekannte zurückführen.

Bei der Erstellung eines Algorithmus wählen Sie die beste Darstellung Ihres Problems aus und bestimmen dann den Algorithmen-Typ.

Jetzt können Sie auch die Art der Arbeitsschritte und eine Reihenfolge festlegen. Es empfiehlt sich, den Algorithmus zunächst von Hand durchzugehen, und auf Fehler zu überprüfen. Nehmen Sie dazu erst allgemeine, abstrakte Werte, denn meistens finden sich schon bei den einfachen Durchgängen Fehler. Sind diese beseitigt, können Sie schon mit spezielleren Daten arbeiten. Wiederholen Sie diesen Vorgang solange, bis Ihr Algorithmus fehlerfrei arbeitet. Bei einfach strukturierten Algorithmen ist es dem erfahrenen Programmierer schon jetzt möglich, einen Ablaufplan zu



Flußdiagramm zu Listing 1



Grafische Darstellung des Dreieck - Tetraeder Problems

erstellen, und in einer Programmiersprache zu formulieren. Für Ungeübte ist es jedoch ratsam, den Algorithmus genauer zu untersuchen. Gehen Sie in diesem Fall nach folgendem Schema vor:

Zerlegen Sie den Algorithmus grob in die erforderlichen Arbeitsgänge. Sie haben dann den Vorteil, jeden Arbeitsgang gesondert betrachten und ihn in einer schrittweisen Verfeinerung bis hin zu einzelnen Arbeitsschritten aufgliedern zu können. Betrachten Sie dann jeden Arbeitsschritt für sich, so erfordert es geringe Mühe, ihn in einer der gewählten Programmiersprache nahen, verbalen Darstellung zu beschreiben. Schreiben Sie dazu Ihre Verarbeitungsanweisungen Schritt für Schritt auf, und legen Sie eventuell eine Tabelle für Eingabewerte, Zwischenergebnisse und Ausgabewerte an. Prüfen Sie das so entstandene Programm mit einem Durchgang per Hand nochmals auf fehlerfreie Abarbeitung.

Das folgende Beispiel soll Ihnen das Verständnis für die in diesem Artikel beschriebene Vorgehensweise erleichtern. Beispiel:

Als Problem sei eine unbestimmte Menge von Kugeln gegeben. Mit diesen Kugeln können Sie sowohl Dreiecke legen als auch Tetraeder bauen. Gesucht sind nun die Mengen von Kugeln, für die folgendes gilt:

Anzahl der Kugeln (Dreieck) = Anzahl der Kugeln (Tetraeder)

Anmerkung: Mit drei Kugeln können Sie ein Dreieck legen, aber keinen Tetraeder bauen. Dazu benötigen Sie vier Kugeln.

Zunächst wird das Problem analysiert. Gegeben ist eine Regel zum Erstellen von Dreiecken und Tetraedern. Gesucht:

K = KD(n) = KT(m)

Am geeignetsten läßt sich das Problem graphisch darstellen (siehe Zeichnung). Wenn Sie die Zeichnung aufmerksam betrachten, stellen Sie für die Dreiecke fest, daß durch Anlegen einer um 1 erhöhten Seite das nächstgrößere Dreieck entsteht. Für die Tetraeder gilt: Aufsetzen des Tetraeders auf das zuvor entwickelte Dreieck ergibt einen neuen Tetraeder. Aus dieser Erkenntnis können folgende (Rekursions-)Formeln abgeleitet werden:

KD(n) = KD(n-1)+n KT(m) = KT(m-1)+KD(m)

Daraus kann man für die Reihenfolge schließen, daß zuerst die Dreiecke und dann die Tetraeder berechnet werden müssen. Aufgrund der gefundenen Formeln können Sie dazu übergehen, einen Algorithmus zu erstellen. Da die Formeln noch nicht näher untersucht worden sind, können Sie nur einen rekursiven Algorithmus aufstellen. Er läßt sich in folgenden Arbeitsgängen beschreiben:

— Berechne Dreiecke — Berechne Tetraeder — Vergleiche alle Dreiecke mit allen Tetraedern.

Damit ist es nun möglich, den Algorithmus per Hand durchzugehen und zu testen. Hilfreich ist es, eine Tabelle für die Ein- und Ausgabewerte anzulegen und die Werte zu vergleichen. Für die ersten fünf Dreiecke und Tetraeder ergeben sich folgende Werte.

n	KD(n)	m	KT(m)
1	1	1	1
2	3	2	4
3	6	3	10
4	10	4	20
5	15	5	35

Das erste Ergebnis wurde schon gefunden:

KD(n=4) = KT(m=3)   K = 10

Aus den Werten in der Tabelle läßt sich aber noch mehr ersehen. Die Dreiecke brauchen nicht mit allen Tetraedern verglichen zu werden. Für die verlangte Problemlösung genügt es, das Dreieck nur solange mit den Tetraedern zu vergleichen, solange KD <= KT ist. Es müssen also nur die Tetraeder berechnet werden, für die diese Bedingung erfüllt ist. Daraus resultiert eine Verkürzung der Rechenzeit, und Speicherkapazität wird ge-

```
 4'tes Dreieck = 3'ter Tetraeder ! Anzahl der Kugeln : 10
 15'tes Dreieck = 8'ter Tetraeder ! Anzahl der Kugeln : 120
 55'tes Dreieck = 20'ter Tetraeder ! Anzahl der Kugeln : 1540
 119'tes Dreieck = 34'ter Tetraeder ! Anzahl der Kugeln : 7140
Programm beendet - keine weiteren Werte gefunden !
```

Testlauf zum Listing

```
10 FOR N=2 TO 150
20   FOR M=1 TO (N-1)
30     KD=(N*(N+1))/2
40     KT=((M*(M+1))*(M+2))/6
50      IF KT>KD THEN 80
60     IF KD=KT THEN PRINT N;"'tes Dreieck =";M;
        "'ter Tetraeder ! Anzahl der Kugeln :";KD
70   NEXT M
80 NEXT N
85 PRINT "Programm beendet - keine weiteren Werte gefunden !"
90 END
```

Listing 1 : Das erwähnte Beispiel in Amiga - Basic

spart. Zu einem guten Programmierstil gehört es, solche Verbesserungen, wo immer möglich, im Programm vorzunehmen.

Diese Vorgehensweise läßt sich mit rekursiven Algorithmen nicht realisieren. Es wäre zwar eine elegante, aber auch zeitaufwendige und speicherintensive Lösung. Aus den Rekursionsformeln erhält man nach den Regeln der Mathematik durch Summenbildung folgende Formeln:

$$KD(n) = \frac{n * (n+1)}{2}$$

$$KT(m= \frac{m * (m+1) * (m+2)}{6}$$

Mit diesen Formeln können Sie nun einen iterativen Algorithmus erstellen. Bauen Sie dazu noch die vorher beschriebenen Laufzeitverbesserungen ein, erhalten Sie einen Algorithmus mit folgenden Arbeitsschritten:

— Berechne ein Dreieck
— Berechne mehrere Tetraeder, solange KT > KD
— Vergleiche Dreieck mit allen Tetraedern
— Wenn KD(n) = KT(m), gebe K,n und m aus
— Berechne nächstes Dreieck, mache weiter mit Schritt 2
— Ende, wenn n = 150

Nachdem sichergestellt worden ist, daß der Algorithmus fehlerfrei arbei-tet, können Sie einen Ablaufplan erstellen und das Programm codieren.

Ich hoffe, Ihnen in diesem Artikel den Weg von einem gegebenen Problem zur Erstellung des Programms ausreichend geschildert zu haben. Das hier vorgestellte Verfahren ist natürlich keine Garantie dafür, ein gestecktes Ziel zu erreichen. Auch ist diese Vorgehensweise nicht bei allen Problemen erforderlich. Sie hilft Ihnen aber, eine gegebene Problematik zu konkretisieren. Die Zeit, die Sie hierfür aufwenden, sparen Sie bei der Fehlersuche im fertigen Programm.

VON ROLAND FÖRSTER

ZUFALLS GENERATOREN

DEM ZUFALL IN DIE KARTEN GESCHAUT

Heute sind Zufallsgeneratoren in fast allen Programmiersprachen implementiert. Sie werden meist über Randomize eingeschaltet, und die Zufallswerte können dann aus der Variablen RND abgerufen werden. Doch wozu benutzt man überhaupt Zufallswerte?

Ein Einsatzgebiet ist die Atomphysik, wo man die Kollision von Atomkernen simuliert, um dann Hypothesen und Berechnungsformeln aufstellen zu können. Im Bereich der Simulation findet man viele Beispiele, die Zufallswerte benötigen. Beispielsweise werden Betriebssysteme, bevor sie auf den Markt kommen, einem Test unterzogen, der Aussagen darüber machen soll, ob das System akzeptable Anwortzeiten unter normalen (zufälligen) Bedingungen liefert. Hierzu wird eine Menge von Tasks {T1,T2,...} verschiedener Größe erzeugt (zufällig), die dann in zufälligen Zeitintervallen zur Bearbeitung abgeschickt werden. In der Betriebswirtschaftslehre kann man mit Hilfe von Zufallszahlen das Käuferverhalten simulieren und somit neue Marketingstrategien entwickeln. Solche Verfahren bezeichnet man als Monte-Carlo-Methoden.

Im Bereich der künstlichen Intelligenz kann man bei komplexen Problemen Zufallsgeneratoren zur Entscheidungsfindung einsetzen. Zum Schluß muß man wohl auch noch das weite Anwendungsfeld der Spielprogrammierung erwähnen, wo Zufallszahlen zur Simulierung von Würfeln, Kartenspielen, Roulettetellern bis hin zu komplizierten Bewegungsverfahren von Asteroiden oder sonstigen grünen Männchen gebraucht werden. Doch wie kann man mit einem Rechner, der immer nach einem Algorithmus vorgeht, Zufallszahlen "berechnen"?

Pseudozufall

Man wird wohl nie wirklich zufällige Zahlen erhalten, denn den Zufall kann man wohl selbst mit dem besten Computer nicht berechnen. Deshalb spricht man auch häufig von Pseudozufallsgeneratoren und Pseudozufallszahlen. Im folgenden sprechen wir also immer von solchen Pseudogrößen. Ein guter Zufallsgenerator muß so programmiert sein, daß man der mit ihm erzeugten Zahlenfolge nicht sofort ansieht, daß sie berechnet und nicht "gewürfelt" wurde.

Woher nehmen ?

Wir wollen uns hier mit dem Problem der Erzeugung uniform verteilter Zufallsgrößen beschäftigen. Eine Zufallsvariable heißt uniform oder gleichverteilt, wenn jede Zahl, die sie annehmen kann, die gleiche Wahrscheinlichkeit hat. Weiterhin sollen die Zufallswerte unabhängig sein, d.h. ein Wert hängt nicht von irgendeinem anderen der erzeugten Werte ab.

Ein Beispiel für eine unabhängig uniform verteilte Zufallsgröße ist ein Würfel. Die Zufallsvariable kann die Werte von 1 bis 6 annehmen, wobei jeder Wert die Wahrscheinlichkeit von 1/6 hat und nicht durch eine andere, schon geworfene Zahl, vorherbestimmt ist. Aus uniform verteilten Zufallszahlen kann man durch geeignete Umformung auch anders verteilte Zufallszahlen wie zum Beispiel standardnormalverteilte oder geometrischverteilte Zufallszahlen machen.

Das 'middle square'-Verfahren

Das erste Verfahren ist unter dem Namen 'middle square' bekannt und wurde von J.v.Neumann 1946 angegeben. Möchte man z.B. eine sechsstellige Zufallszahl erzeugen, so gibt man einen beliebigen sechsstelligen Startwert vor, beispielsweise das heutige Datum. Dieser Startwert wird nun quadriert. Die mittleren sechs Stellen der Quatratzahl bilden die neue Zufallszahl.
Dieses Verfahren wird dann iterativ fortgesetzt (siehe Listing 1).

```
10 REM middle sqare generator
20 INPUT Z#
30 FOR I=1 TO 20
40   Z#=Z#*Z#
50   W$= STR$(Z#)
60   W$= MID$(W$,5,6)
70   Z#= VAL(W$)
80 PRINT Z#
90 NEXT
100 END
```

Listing 1: Der Middle-Square-Generator

Die 'middle square'-Methode ist kein besonders gutes Verfahren, da sie sehr häufig zu dem Wert 0 degeneriert, wenn man zu kleine Startwerte wählt. Besser wäre es, eine zehnstellige anstatt einer sechsstelligen Zahl als Anfangswert zu wählen. Die zweite Methode liefert da schon wesentlich bessere Ergebnisse.
Mit ihr erhält man Zufallsgrößen zwischen 0 und 1. Aus diesen kann man jedoch ohne Probleme jeden gewünschten Bereich durch Streckung erhalten. Möchte man zum Beispiel Lottozahlen erzeugen, kann man das durch folgende Umrechnung erreichen: Es sei Z die Zufallsvariable mit Werten aus dem Bereich zwischen 0 und 1, dann ist

$$L = INT(Z * 49) + 1$$

eine uniformverteilte Zufallsgröße in dem Intervall 1 bis 49. Für einen Würfel sähe die Formel dann so aus:

$$L = INT(Z * 6) + 1$$

Doch nun zu dem Verfahren, mit dem man Z erzeugen kann. Es hat etwas mit der Kreiszahl PI zu tun. Als Grundwert muß diesmal eine Zahl eingegeben werden, die kleiner als 1 und größer als 0 ist. Zu dieser Zahl wird dann PI (3.141592654) addiert. Die so erhaltene Zahl wird acht mal mit sich selbst multipliziert, und anschließend werden nur die Dezimalstellen betrachtet. Die somit erhaltene Zufallszahl dient als Ursprung für weitere Berechnungen. Siehe auch Listing 2 und das Beispielprogramm, das einen Lotto Tip erzeugt.

Lineares Kongruenz Schema

Das letzte Verfahren, das ich vorstellen möchte, ist unter dem Namen "Lineares Kongruenz Schema" bekannt. Es ist das beste, aber auch das mathematisch komplizierteste. Es basiert auf der Formel

```
10 REM Zufallsgenerator 2
20 INPUT Z
30 FOR I=1 TO 20
40   Z=Z+3.141
50   Z=Z^8
60   Z=Z- INT(Z)
70   PRINT Z
80   X=Z
90 NEXT I
100 END
```

Listing 2: PI kommt ins Spiel

$$X(n+1) = (a * X(n) + C) \bmod m$$

Die neue Zufallszahl wird aus der letzten berechnet, indem man sie mit einer Konstanten a multipliziert, dann einen festen Wert C dazu addiert und schließlich die Restklasse modulom betrachtet. Unter Restklasse versteht man die ganzzahligen Reste, die bei einer Division durch m auftreten können. Beispiele:

12 mod 5 = 2
17 mod 2 = 1

Zu suchen sind vier Werte, mit denen das Verfahren optimal verteilte Zahlen liefert. Wählt man beispielsweise X(0) = a = c = 7 und m = 10, so werden nur die Zahlen 7, 6, 9, 0, 7, 6, 9, 0, ... erzeugt. Das heißt, das Verfahren produziert eine endlose Reihe dieser vier Zahlen, wobei die Periodenlänge vier ist. Doch wie soll man die Werte geeignet bestimmen ? m sollte möglichst groß sein, da man dadurch auch eine große Periodenlänge erwarten kann. Es sollte aber auch nicht zu lange dauern, bis die Zufallszahl berechnet ist, deshalb ist es günstig, m kleiner als die Wortlänge des Rechners zu wählen (Rechenzeit). Eine Möglichkeit besteht nun darin, m gleich der größten Primzahl kleiner zwei hoch der Wortlänge zu setzen. Zur Berechnung dieser Zahl kann man z.B. das Sieb des Erasthostenes nehmen, welches auch sehr oft als Benchmark-Programm benutzt wird. Das additive Glied C kann man gleich 0 setzen, was jedoch den Nachteil hat, daß die Periodenlänge etwas schrumpft. Die Wahl des Faktors ist eine außerordentlich komplizierte Angelegenheit, und ich empfehle, mal ein bißchen zu probieren. Als Beispiel soll das dritte Listing dienen.

Weiterführende Literatur:
Donald E. Knuth
The art of computer programming
Vol 2 / seminumerical algorithms
1969 Addison Wesley

```
10 REM Lottotip
20 DIM Lotto(6)
30 INPUT"seed ",Z
40 REM 6 verschiedene
   Lottozahlen bestimmen
50 FOR I=1 TO 6
60   GOSUB 220
70   Lotto(I)= INT(Z*49)+1
80   FOR J=1 TO I-1
90     IF Lotto(I)=Lotto(J)
       THEN GOTO 60
100   NEXT J
110 NEXT I
120 REM Ausgewaehlte
    Lottozahlen mit dem
    Bubble-sort sortieren
130 FOR I=1 TO 6
140   FOR J=6 TO I+1 STEP -1
150     IF Lotto(I)>Lotto(J)
    THEN A=Lotto(I):Lotto(I)=
    Lotto(J):Lotto(J)=A
160   NEXT J
170 NEXT I
180 FOR I=1 TO 6
190   PRINT Lotto(I);
200 NEXT
210 END
220 REM Zufallsgenerator 2
230   Z=Z+3.141
240   Z=Z^8
250   Z=Z- INT(Z)
260   X=Z
270 RETURN
```

Listing 3: Der Tip der nächsten Woche

VON GERALD CARDA

C-KURS

Teil 5

Der fünfte Teil des C-Kurses wird sich mit Zeigern oder auch Pointern beschäftigen. Zeiger und der Umgang mit Zeigern sind eine Spezialität von C. Viele Probleme können so leicht programmiert werden. Es gibt jedoch auch nichts Einfacheres, als sich bei der Programmierung von Zeigern so zu vertun, daß schließlich garnichts mehr geht. Sie sollten also dieser Folge Ihre ganze Aufmerksamkeit widmen und im sieZweifelsfall noch einmal durchgehen.

Zeiger sind eine besondere Art von- Variablen. In ihnen werden keine Werte gespeichert, sondern Adressen, unter denen dann die Werte zu finden sind. Sie können sich dazu alle Speicherzellen Ihres Rechners durchnumeriert denken. In jeder Speicherzelle kann eine Variable enthalten sein. Ist beispielsweise in der Speicherzelle Nummer 10 die Variable A enthalten, ist die Adresse der Variablen A gleich 10. Folgende Tabelle soll dies noch einmal verdeutlichen:

Adresse	Variable	Wert
00	X	2222
01	D	1234
02	M	6785
10	A	666

Wie Sie sehen, ist der Wert der Variablen A gleich 666 und ihre Adresse gleich 10. Es ist sehr wichtig, daß Sie sich bewußt machen, daß die Adresse und der Wert zwei unterschiedliche und streng zu trennende Attribute einer Variablen sind. Die Adresse einer Variablen spiegelt die Lage im Speicher wieder, der Wert einer Variablen ist die an dieser Stelle gespeicherte Information.

Um in der SpracheC mitZeigern arbeiten zu können, müssen diese erst, wie andere Variablen auch, definiert werden. Dies geschieht an derselben Stelle wie auch für die anderen Variablenarten - entweder innerhalb einer Funktion oder aber ganz am Anfang eines Programmes als globale Variablen.

Zeiger enthalten zu ihrer Kennzeichnung einen Stern vor dem Variablennamen. Die nächste Zeile zeigt Ihnen die Deklaration von einigen Zeigern.

```
int *zeigzahl;
char *zeigbuchstabe;
long * zeig große zahl;
```

Hier wurden gleich drei "verschiedene" Arten von Zeigern deklariert. Der Zeiger 'zeigzahl' ist als Zeiger auf eine Variable des Types Integer deklariert, die beiden anderen als Zeiger auf einen Char und einen Long Wert.

Rein programmiertechnisch unterscheiden sich Zeiger auf verschiedene Arten von Datentypen nicht voneinander. D.h., daß alle Operationen, die auf Zeiger anwendbar sind, auch mit allen Arten von Zeigern arbeiten. Für den Compiler ist es nämlich wichtig zu wissen, ob die Variable eine Zahl, einen Buchstaben oder einen Langwert enthält, auf die der Zeiger zeigt. Das liegt in der Tatsache begründet, daß alle diese Variablentypen eine andere Größe im Speicher haben. Ein Integer Wert kann z.B. 16 oder 32 Bit groß sein, ein Char-Wert ist jedoch nur 8 Bit groß. Der Compiler prüft, ob der Programmierer nicht aus Versehen zwei unterschiedliche Zeigerarten vermischt. Diese Option ist jedoch auch abschaltbar, so daß Sie mit den Zeigern so frei umgehen können, wie Sie wollen.

Die Sprache C läßt Ihnen in diesem Punkt sehr große Freiheiten. Sie können mit Zeigern rechnen wie mit ganzen Zahlen. D.h., Sie können zu Zeigern Zahlen addieren oder subtrahieren. Schauen wir uns dazu einmal ein kleines Beispiel an.

```
int a_null;
int a[10];
int *zeig_a;
zeig_a = &a[0];
a_null = *zeig_a;
```

In diesem kurzen Programmstück passiert schon ganz schön viel Neues. Die ersten drei Zeilen müßten Ihnen noch leicht verständlich sein. Hier werden einige Variablen deklariert. Zuerst einmal eine Variable vom Typ Integer und ein Array mit 10 Elementen. Dann folgt die Deklaration eines Zeigers auf einen Integer - Wert.
Die dann folgende Zeile enthält gleich zwei Neuigkeiten. In ihr wird der Zeigervariablen 'zeig_a' die Adresse des Arrayelementes 'a[0]' zugewiesen. Wie Sie sehen, ist dazu eine besondere Schreibweise nötig. Betrachten wir zuerst die Syntax bei der Zuweisung von Adreßwerten an Zeigervariablen. Soll einer Zeigervariablen eine Adresse zugewiesen werden, ist der Variablenname des Zeigers ohne vorangestellten Stern zu verwenden. Den Wert einer Adresse, an der eine beliebige Variable gespeichtert ist, erhält man durch die Verwendung des Adreßoperators '&'. Folgende Tabelle zeigt Ihnen noch einmal die Auswirkungen der verschiedenen Möglichkeiten, die sich bei der Verwendung der Operatoren '*' und '&' ergeben.

```
int platzhalter;
int *zeiger;
```

platzhalter liefert den Wert, der in der Variablen 'Platzhalter' gespeichert ist.

```
&platzhalter
```

Liefert die Adresse, in der der Wert der Variablen'Platzhalter' gespeichert ist.

```
zeiger
```

Liefert die Adresse, auf die der Zeiger momentan zeigt.

```
*zeiger
```

Liefert den Wert, der in der Adresse gespeichert ist, auf den der Zeiger momentan zeigt.

Kommen wir jetzt zu einer interessanten Möglichkeit, die die Programmierung mit Zeigern eröffnet. Hierzu kommen wir noch einmal auf das Array zurück. Arrays und Zeiger sind in C sehr stark miteinander verbunden. Man kann eigentlich alle Zugriffe auf ein Arrayelement auch über Zeiger abwickeln. Die folgende Aufstellung zeigt Ihnen den jeweils äquivalenten Befehl.
Wir gehen wieder von einem Array mit 10 Elementen aus.

```
int a[10];

zeig_a = &a[0];
```

Nach diesen Vorbereitungen haben wir einen Zeiger, der auf das erste Element im Array 'a' zeigt.

```
xyz = a[0];
```

Der Variablen 'xyz' wird der Wert von 'a[0]' zugewiesen. Folgende Anweisung würde genau das gleiche bewirken.

```
xyz = *zeig_a;
```

Der Zugriff auf weitere Elemente sieht wie folgt aus :

```
xyz = a[1];
xyz = *(zeig_a + 1);
xyz = a[5];
xyz = *(zeig_a + 5);
usw.
```

Diese Art des Zugriffs auf Variablen über Zeiger und die Addition eines Offsets ist keineswegs so selbstverständlich, wie es in diesem Beispiel aussieht. Dem aufmerksamen Leser ist sicherlich eine kleine Unstimmigkeit in den eben beschriebenen Zugriffsmöglichkeiten auf Variablen über Zeiger aufgefallen.
Wir wenden uns dazu noch einmal der Darstellung von Variablen im Speicher zu. Das Array 'a' sei ab der Speicheradresse 1000 im Arbeitsspeicher des Rechners abgelegt. Die dort gespeicherten Werte sind in diesem Beispiel durch vierstellige Buchstabenkombinationen dargestellt.

Adresse	Wert
1000	aaaa
1002	bbbb
1004	cccc
1006	dddd
1008	eeee
1010	ffff
.	.

In diesem Beispiel wird von einer Zahlengröße von 16 Bit - das entspricht zwei Speicherzellen mit je einem Byte - ausgegangen. Das erste Element belegt also die Speicherzellen 1000 und 1001. Das zweite liegt dann entsprechend in den Adressen 1002 und 1003.
Durch die Zuweisung 'zeig_a = &a[0]' erhält die Zeigervariable 'zeig_a' den Wert 1000 zugewiesen. Wenn wir jetzt die obige Anweisung 'xyz = *(zeig_a + 1)' wollen, müßte doch eigentlich folgendes passieren :

```
zeig_a = 1000
*zeig_a = aaaa

zeig_a + 1 = 1001

*(zeig_a + 1) = aabb  ????
```

Nach dieser Überlegung würde also das oben gezeigte Beispiel nicht funktionieren. Daß es in C doch arbeitet, liegt an der speziellen Zeigerarithmetik des C-Compilers. Diese Arithmetik sogrt dafür, daß bei der Addition von 1 zu einem Zeiger nicht wirklich der Wert 1 addiert wird, sondern die Anzahl von Speicherzellen, die die Variablenart groß ist, auf die der Zeiger zeigt. Dasselbe gilt auch für den Operator ++, der eine Variable hochzählt.
Bei den ganzen in dieser Folge gezeigten Zuweisungen von Zeigern auf andere Variablen ist eine Besonderheit zu beachten. Wir betrachten dazu noch einmal folgende Zuweisung :

```
zeig_a = &a[0];
```

Diese Zuordnung kann auch kürzer geschrieben werden :

```
zeig_a = a;
```

Top aktuell:



Amiga Tips & Tricks – jetzt in einer völlig überarbeiteten Neuauflage. Hier verraten Ihnen echte Profis, mit welchen Tricks sie mehr aus dem Amiga holen: Hilfen zur Gestaltung eigener Programme, Tips & Tricks zum AmigaBASIC, Maschinenprogramme in Amiga-BASIC, Einsatz von DOS-Routinen, optimierende Hilfsprogramme für AmigaBASIC-Programme, Tips zur Arbeit mit der Workbench, Aufbau der Icons, neue Ein-/Ausgaberoutine. Mit vielen Anregungen, aber auch fertigen Lösungen. Greifen Sie in die Trickkiste, und schon werden Dinge wahr, die Sie nicht für möglich hielten. Ein Buch, das voller Überraschungen steckt.
Amiga Tips & Tricks – die riesige Fundgrube für jeden Amiga-Besitzer.

Amiga Tips & Tricks
Hardcover, 473 Seiten
DM 49,–

Der Amiga macht es einem so leicht wie möglich. Nahezu alles läßt sich problemlos über die Workbench bearbeiten. Wenn Sie jedoch den Mut haben, die komfortable Oberfläche zu verlassen, werden Sie schon sehr bald belohnt – mit einigen Dingen, die Sie dem Amiga bisher nicht zugetraut hätten. Das große Buch zu AmigaDOS hilft Ihnen dabei. Neben einem ausführlichen Einsteigerteil erfahren Sie alles, was Sie bei Ihrer praktischen Arbeit mit dem AmigaDOS wissen sollten: Umlenken der Ein- und Ausgabe, sinnvoller Einsatz des Jokers, Arbeiten mit RAM-Disk und CLI, Batch-Dateien, STARTUP-Sequenz, Multitasking mit dem CLI, Aufbau der CLI-Befehle, Programmierung eigener CLI-Befehle, neue CLI-Befehle in BASIC und C. Dazu ein ausführlicher, gut strukturierter Nachschlageteil. Wer also mit dem AmigaDOS arbeiten möchte, sollte dieses Buch immer in greifbarer Nähe haben.

Das große Buch zu AmigaDOS
Hardcover, 320 Seiten
DM 49,–

Das Buch, das zur Amiga-Floppy keine Frage offenläßt. Hier finden Sie Dinge, die Sie im Handbuch vergeblich suchen werden: Floppy-Operationen unter der Workbench und unter AmigaDOS im CLI, relative und sequentielle Dateien, Aufbau der Diskette, Zugriff über Trackdisk-Device, Track lesen und schreiben, Kodier-und Dekodierroutinen des Betriebssystems… Mit vielen nützlichen Programmen wie z. B. ein Superkopierprogramm oder einen Floppyspeeder. Was Sie wissen müssen, finden Sie hier – vom Einsteiger zum Profi.

Amiga-Floppy-Buch
Hardcover, ca. 350 Seiten
inkl. Diskette, DM 59,–
erscheint ca. 2/88

DATA BECKER
Merowingerstr. 30 · 4000 Düsseldorf · Tel. (0211) 310010

BESTELL-COUPON
Einsenden an: DATA BECKER · Merowingerstr. 30 · 4000 Düsseldorf 1
Bitte senden Sie mir:

zzgl. DM 5,– Versandkosten
unabhängig von der bestellten Stückzahl
☐ per Nachnahme ☐ Verrechnungsscheck liegt bei

Name
Straße
Ort

Die Ähnlichkeit geht sogar noch etwas weiter :

```
xyz = a[5];

entspricht

xyz = *(a+5);
```

Eine Operation, die hier jedoch nicht erlaubt ist, ist das Verändern des Wertes von 'a', z.B. durch folgende Zeile :

```
a = zeiger_auf_b;

oder

a++;
```

Der Bezeichner eines Arrays, auch Vektor genannt, ist in C eine Konstante und kann, im Gegensatz zu Zeigervariablen, nicht verändert werden. Man kann dies jedoch umgehen, indem man den Vektor an eine Funktion übergibt, die intern mit Zeigern arbeitet. Das folgende kleine Programm zeigt eine kleine Funktion, die die Länge einer Zeichenkette bis zum ersten Leerzeichen zählt.

```
zaehle_bis_erstes_space(s)
char *s;
{
  int n;
  for(n=0;*s!=' ';s++)
     n++;
  return(n);
}
```

An die Funktion wird ein Vektor einer Zeichenkette übergeben, der intern jedoch als Zeiger interpretiert wird. Auf diese Weise kann die verlangte Funktion sehr leicht programmiert werden. Man findet diese Vorgehensweise sehr häufig in C-Programmen.
Ich möchte Ihnen an dieser Stelle erst einmal die Gelegenheit geben, das neue Wissen zu verdauen. Die Programmierung mit Zeigern ist in C eine sehr wichtige Sache und sollte daher von Ihnen einiges an Aufmerksamkeit bekommen.
Viel Spaß und bis zum nächsten Mal.



VON SEBASTIAN DOSCH

TIPS & TRICKS ZU DELUXE PAINT II

Teil 3: Wie wäre es mit einem eigenen Spiel?



Bild 1: Bevor es losgeht, muß zunächst das Spielfeld abgesteckt werden

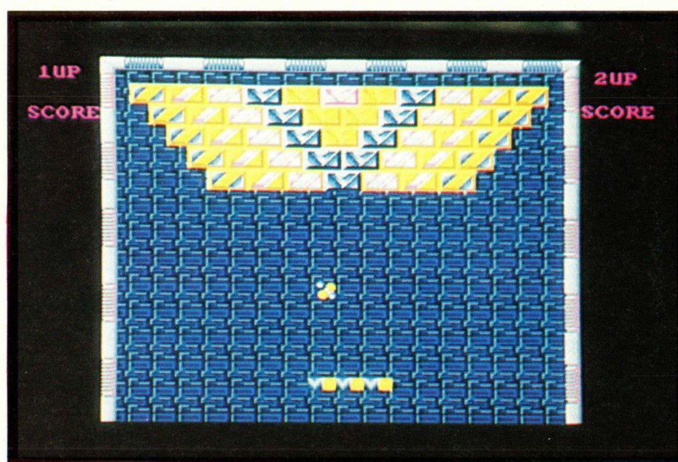


Bild 2: Ein ausgefüllter Hintergrund

In dieser Ausgabe möchte ich die Gestaltung eines Break-Out-Spiels behandeln. Diese Art von Spiel hat in letzter Zeit immer mehr an Popularität gewonnen. Bei seiner Programmierung auf dem AMIGA spielt gerade die Grafik eine große Rolle. Doch dazu braucht man ein gutes Hilfsprogramm. Was liegt also näher, als DPAINT zu Hilfe zu nehmen?

Sollten Sie sich entschlossen haben, für dieses oder ein anderes Spiel die Grafik zu zeichnen, dann sollten Sie sich zunächst den Namen des Spiels, den Screenaufbau und das Geschehen rund um das Spiel ausdenken. Ich werde in diesem Kurs die Dinge in einer Reihenfolge erklären, die Sie immer einhalten sollten. Wir werden mit dem Hintergrund, der Score-Leiste und den Anzeigen beginnen und dann mit den Steinen und dem Schläger fortfahren. Schließlich widmen wir uns den herumfliegenden Sprites oder Bobs, der Highscoreliste und zeichnen erst ganz zum Schluß das Titelbild.

1. Die Score-Leiste

Hier müssen Sie sich zuerst einmal überlegen, wie Ihre Score-Leiste aussehen soll. Darin enthalten sein sollten auf jeden Fall der Name des Spiels, die Anzeige der Punkte, die Höchstpunktzahl und die Anzahl der teilnehmenden Spieler. Bei vorhandenen Special-Funktionen (wie z.B. zwei Bälle, gespiegelte Lenkung des Schlägers, doppelt breiter Schläger, Abschießen von Steinen, magnetischer Schläger usw.) darf die Anzeige der gerade aktiven Funktion natürlich nicht fehlen. Na, da kommt doch schon eine ganz schöne Palette zusammen, oder nicht?

Ich würde Ihnen empfehlen, das Ganze unter- oder nebeneinander gegliedert auf dem Bildschirm darzustellen. Beachten Sie dazu Beispiel. Sie könnten aber auch zu der Score-Leiste pas-

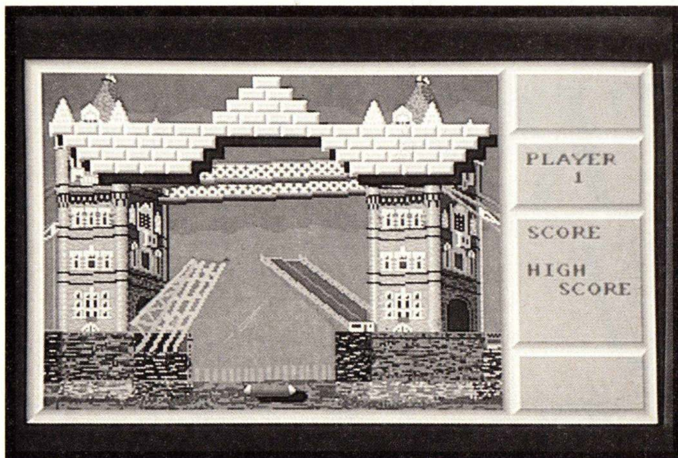
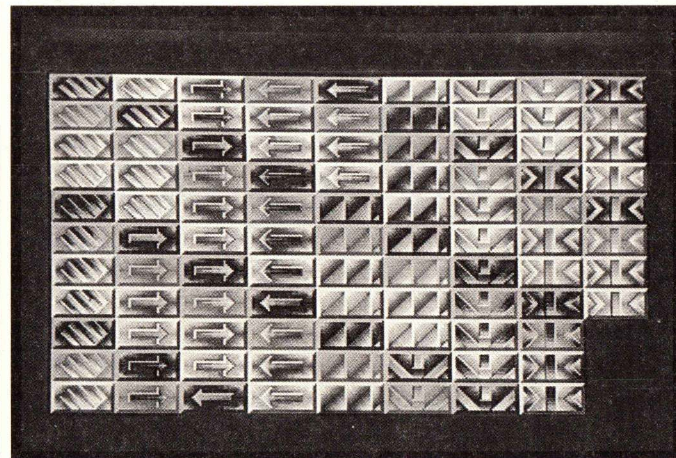


Bild 3: geradezu ideal ist ein Bild als Hintergrund


Bild 4; Einfarbige Spielsteine sind dem AMIGA nicht würdig

send einen Rand um das Spielfeld herum aufbauen.

So, jetzt hätten wir ja schon ein schönes Aussehen für unseren Screen. Aber soll der Untergrund, auf dem man später spielt, etwa schwarz bleiben? Stellen wir stattdessen doch ein schönes Muster oder eine Hintergrundgrafik darin dar. Für die Muster nehmen wir am besten Patterns, die mit DPaintII gezeichnet werden (Bild 2). Achten Sie aber bei den Mustern darauf, daß sie immer aneinander passen, und verwenden Sie keine allzu grellen Farben. Heben Sie sich diese für die Steine auf, denn anderfalls wird der Spieler während des Spielens sehr beeinträchtigt.

Und noch etwas: Finden Sie es nicht auch schön, eine von Steinen bedeckte Landschaft oder ein Bild freizulegen? Ein Beispiel hierzu sehen Sie in Bild 3. Man erkennt sofort, daß das einen besonderen Reiz haben kann. Wem der ruhende Untergrund nicht genügt, der sollte ihn scrollen lassen.

Was halten Sie von diesen Ideen? Ich hoffe, daß für jeden Geschmack etwas dabei war.

Kommen wir nun zu den wichtigsten Bestandteilen eines Break-Out-Spieles: dem Schläger und den Steinen.

2. Schläger & Steine

Überlegen Sie sich nun, wie groß und in welchen Farben die Steine später auf dem Bildschirm dargestellt werden sollen. Und denken Sie auch über Funktionssteine wie z.B. Bonusleben, Bonuspunkte usw., die evtl. in das Spiel eingebaut werden sollen, nach.

Zeichnen Sie nun die Steine in der von Ihnen gewählten Größe (nicht zu klein!) auf den Bildschirm. Aber bitte keine einfarbigen und wirklich öde aussehenden Steine wählen, lassen Sie ihrer Phantasie ruhig freien Lauf.

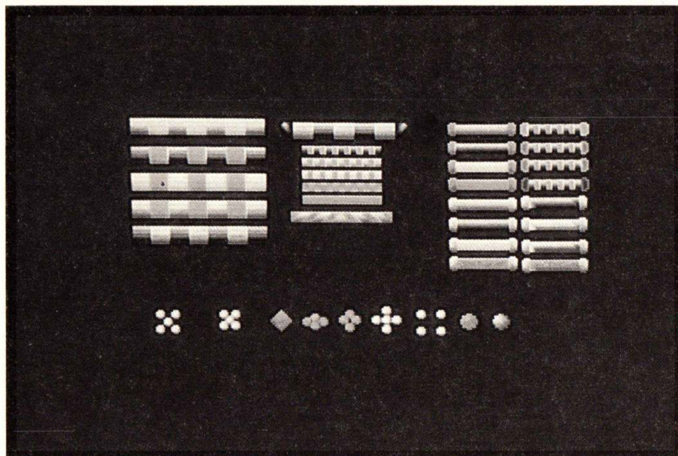
Was Sie noch beachten sollten, ist, eine Licht- und eine Schatten-Seite auf den Steinen zu erzeugen, denn das gibt ihnen erst die richtig plastische Wirkung. Bei der Schlägergröße empfehle ich die Breite zweier nebeneinander gesetzter Steine, da sich diese als angenem erwies.

Lassen Sie sich aber bitte auch hier etwas Originelles einfallen (siehe Bild 4). “Aber wo ist denn der Ball, mit dem man spielen soll?”, werden manche von Ihnen sich jetzt fragen. Dieses Thema werden wir natürlich sofort behandeln, aber beachten Sie dazu doch bitte zuerst einmal Bild 5. Dort sehen Sie einige Vorschläge für Bälle, die etwas anders aussehen als weiße Kreise mit einem roten Punkt in der Mitte.

Oder was halten Sie von einem animierten Ball? Da macht später das Spielen gleich viel mehr Spaß.

3. Objekte

Ich möchte Sie nicht unbedingt beeinflussen bei der Wahl zwischen der Gattung “Sprite” und “Bob” für Ihre herumfliegenden Objekte. Wählen Sie die Ihnen am besten liegende aus und setzen Sie sie in ihr Spiel ein. Aber auch hier appeliere ich an Ihre Originalität. Und achten Sie darauf: die herumwirbelnden Objekte sollen in erster Linie als Bonus und als bewegliches Hindernis für den Ball gelten


Bild 5: Eine kleine Auswahlpalette für Schläger und Bälle.

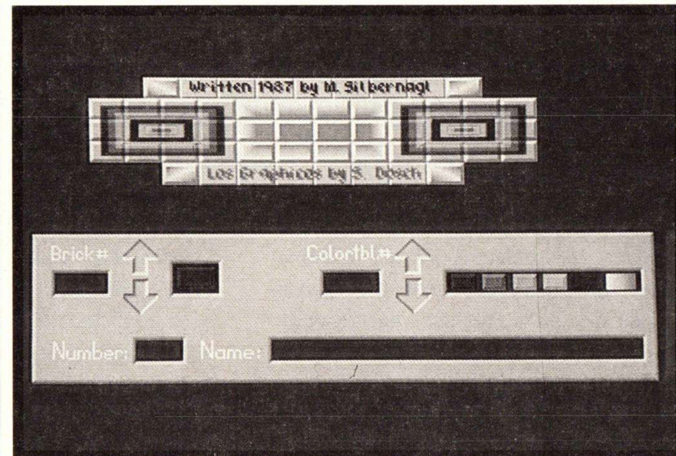


Bild 6: Ein Construction-Kit bietet eine große Hilfe zur Spielentwicklung.

und nicht ein eigenständiges Spiel darstellen.

4. Highscoreliste

Die Highscoreliste sollte übersichtlich gegliedert sein und einen farbenfrohen Eindruck machen. Aber verfallen Sie bitte nicht auf die ordinäre Namenseingabe per Tastatur, diese Zeiten sind nun wirklich vorbei!!! Was halten Sie beispielsweise von einer kleinen Murmelbahn, bei der man die Bahn so einrichten muß, daß der unter dem Röhrchen befindliche Buchstabe von der Murmel getroffen wird?

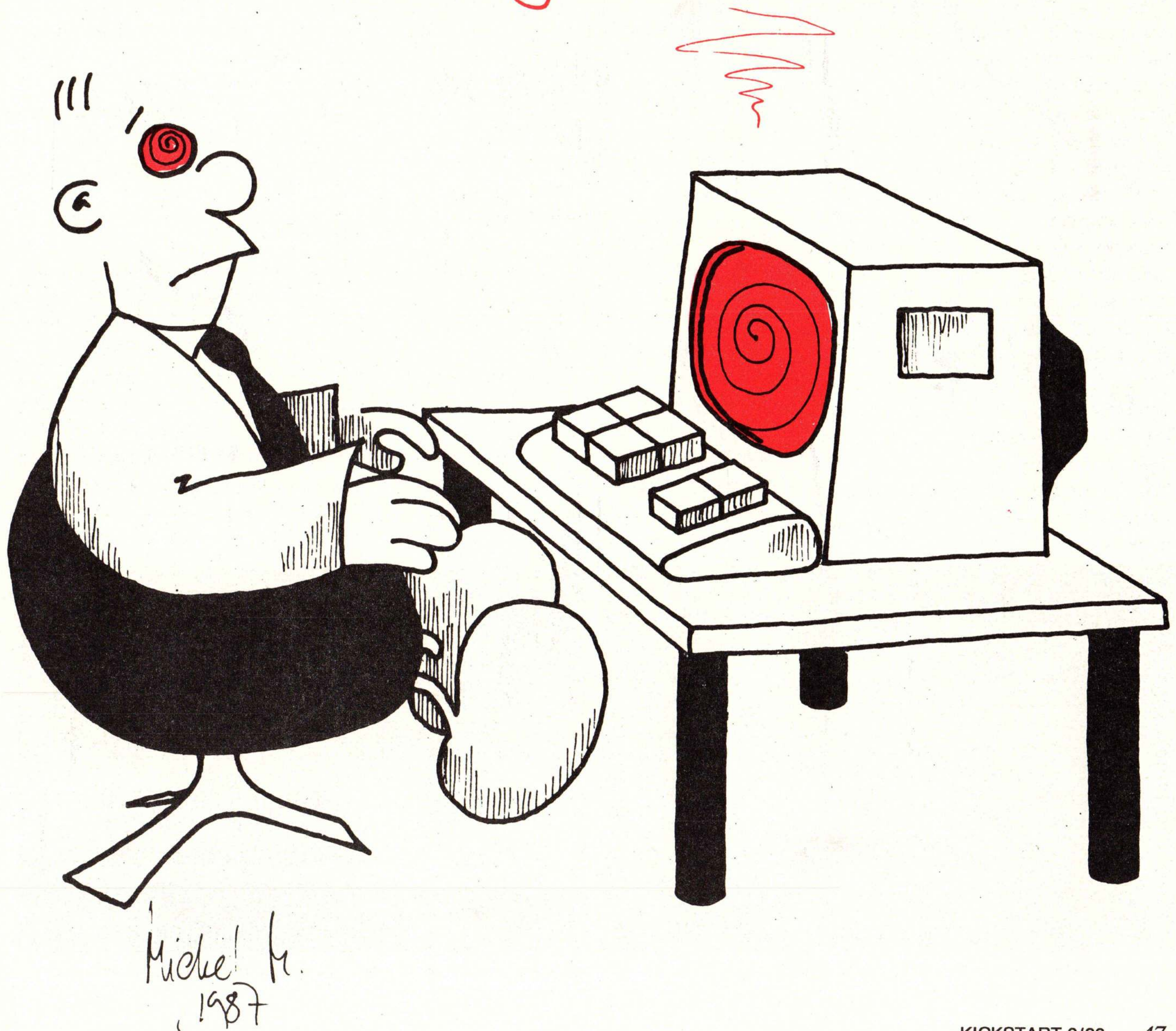
5. Titelbild

Beim Titelbild möchte ich Ihnen nicht viel hineinreden; hier spielt der eigene Geschmack eine große Rolle. Platz für Ihre Kreativität ist genug. Vergessen Sie aber keinesfalls Ihren Namen in diesem Bild zu verewigen.

Zum Schluß noch ein kleiner Tip: Die Stein-Kombinationen lassen sich am besten mit Hilfe eines Construktion-Kits entwerfen. Beachten Sie dazu auch Bild 6 des Construction-Kits zu dem Spiel "Break'em".

Ich hoffe, daß Ihnen dieser Ausflug in die Spielentwicklung weitergeholfen hat, oder Ihnen zumindest einen Eindruck von der Arbeit, die in der grafischen Gestaltung eines Spiels steckt, vermittelt hat.

Deine Augen sind schweeeer – ganz schwer ... Du bist müde ... Du stehst nun vollständig unter meiner Kontrolle!



FÜR ENTSCHLOSSENE



VON HERBERT KUNZ

MIT BASIC ANS SYSTEM

Teil 1: Nutzung der System Routinen von Amiga Basic

Die zum Amiga mitgelieferte Programmiersprache ist Amiga Basic. Der Sprachschatz dieses Basic Interpreters erlaubt es, die meisten der Möglichkeiten, die der Amiga bietet, zu nutzen. So können Pull Down Menüs erzeugt werden, die Maus kann auf Bewegungen und Knopfdrücke abgefragt werden, das Programmieren von Fenstern erfolgt mit einfachen Basic-Befehlen u.s.w. Wer noch mehr aus dem Amiga herausholen will, für den bietet Amiga Basic einen recht leistungsfähigen Befehl, den Befehl LIBRARY.

Fangen wir an ...

Durch diesen Befehl haben Sie die Möglichkeit, die Betriebssystemroutinen des Amiga aufzurufen. In der Regel werden diese Routinen benutzt, wenn Sie Ihren Amiga in C oder Assembler programmieren. Aber auch bei der Programmierung in Amiga Basic kann es recht nützlich sein, über die wichtigsten Systemroutinen Bescheid zu wissen. Aufgabe dieses Kurses soll es sein, Ihnen Einblick in die wichtigsten Libraries zu geben, und anhand von Beispielen die Nutzung einiger Systemroutinen zu erklären.

Die Libraries

LIBRARY heißt auf deutsch Bibliothek. Wenn Sie wissen wollen, welche Bibliotheken Ihnen der Amiga zur Verfügung stellt, dann laden Sie einmal Amiga Basic, legen Sie die Extras-Diskette ein und tippen folgendes ein:

```
CHDIR"df0:fd1.2 "files
```

Es werden nun sämtliche Systemlibraries aufgelistet mit dem Anhang _lib.fd. In Tab. 1 sehen Sie, welche Libraries welche Funktion haben. Wir wollen einmal in diese Libraries hineinschauen. Tippen Sie das kurze Programm LIB.READ ab und starten Sie es. Sie werden aufgefordert, den Namen der Library anzugeben. Geben Sie diesen Namen ohne den Anhang _lib.fd ein. Ihn setzt das Programm automatisch. Es werden nun die einzelnen Systemroutinen mit ihrem Namen aufgelistet. Ein Beispiel sehen Sie in Bild 1 auf der nächsten Seite (Auszug aus der Graphics Library). Hinter dem Namen stehen die Parameter, die bei diesem Befehl mit angegeben werden müssen. Hinter den Parametern stehen die Register, in denen diese Parameter eingesetzt werden müssen. Diese sind aber nur für den Assemblerprogrammierer interessant. Nun können Sie zwar die Libraries lesen, aber Amiga Basic noch nicht. Es benötigt ein spezielles Datenfile, um die Systemroutinen aufrufen zu können.

Die .bmap Dateien - nur für Basic-Programmierer

Es sind dies die Dateien mit dem Anhang .bmap. Einige .bmap-Dateien befinden sich bereits in der Schublade BasicDemos der Extras-Diskette. Aber leider noch nicht alle. Zum Erstellen dieser Dateien dient das Programm ConvertFD auf der Extras-Diskette. Wenn Sie dieses Programm starten, werden Sie aufgefordert den Dateinamen der fd Datei anzugeben. Sie müssen hier den vollen Namen und Pfad angeben; also zum Beispiel für die intuition Library:
df0:fd1.2/intuition_lib.fd
Für die anzulegende .bmap-Datei geben Sie ein: ***intuition.bmap***
Die .bmap Dateien haben folgendes

```
##base _GfxBase
##bias 30
##public
'------ Text routines
BltBitMap(srcBitMap,srcX,srcY,destBitMap,destX,destY,sizeX,sizeY,minterm,mask,tempA)(A0,D0/D1,A1,D2/D3/D4/
D5/D6/D7/A2) BltTemplate(source,srcX,srcMod,destRastPort,destX,destY,sizeX,sizeY)(A0,D0/D1/A1,D2/D3/D4/D5)
ClearEOL(rastPort)(A1)
ClearScreen(rastPort)(A1)
TextLength(RastPort,string,count)(A1,A0,D0)
Text(RastPort,string,count)(A1,A0,D0)
SetFont(RastPortID,textFont)(A1,A0)
OpenFont(textAttr)(A0)
CloseFont(textFont)(A1)
AskSoftStyle(rastPort)(A1)
SetSoftStyle(rastPort,style,enable)(A1,D0/D1)
'------ Gels routines ------
##public
AddBob(bob,rastPort)(A0,A1)
AddVSprite(vSprite,rastPort)(A0/A1)
DoCollision(rasPort)(A1)
DrawGList(rastPort,viewPort)(A1,A0)
InitGels(dummyHead,dummyTail,GelsInfo)(A0/A1/A2)
InitMasks(vSprite)(A0)
```

Auszug aus der Graphics Library

Tab. 1 - Die Libraries und ihre Bedeutung

Icon.Library - Zuständig für die Verwaltung der Workbench Symbole.

Layers.Library - Verwaltet die Überlagerten Bildschirmbereiche.

Exec.Library - Zuständig für Speicher und Multitasking.

Graphics.Library - Die graphischen Funktionen.

Translator.Library - Sprachausgabe.

Console.Library - Textausgabe in Consolen-Fenster (z.B CLI)

Intuition.Library - Screens, Fenster, Gadgets, Menüs, u.s.w.

Timer.Library - zur Programmierung von genauen Zeitintervallen.

Diskfont.Library - Erlaubt die Benutzung der Diskresidenten Zeichensätze.

PotgoLibrary - Für analoge Eingaben.

Expansion.Library - Für Erweiterungen.

Dos.Library - Die Disk Routinen.

Clist.Library - Für die Verwaltung von Copper-Lists.

Mathieeedoubbas.Library - Mathematische Routinen für Integer.

Cstrings.Library - Routinen für Strings.

Mathtrans.Library - Winkelfunktionen, Logarithmusberechnungen.

Mathffp.Library - Mathematische Routinen für Fließkommazahlen

Die Libraries und ihre Bedeutung

Format: Zuerst steht dort der Name der Systemroutine als ASCII-Zeichen, abgeschlossen mit einem Null-Byte. Dann folgt der Offset des Befehls als vorzeichenbehafteter 16-Bit Wert und zum Schluß Zahlenwerte für die Register, in denen die Parameter übergeben werden, abgeschlossen mit einem Null-Byte. Die Werte 1-8 stehen für die Register D0-D7 und die Werte 9-13 für die Register A0-A4. Für unsere Experimente mit den Systemroutinen legen Sie sich am besten eine Diskette an, auf die Sie die .bmap-Dateien kopieren.

Gewußt wie !

Wie eingangs erwähnt, wird eine Systembibliothek mit dem Befehl Library zugänglich gemacht. Hinter dem Library-Befehl steht der Name der Bibliothek in Anführungszeichen - aber ohne den Anhang .bmap, dafür mit dem Anhang .library. Wenn Sie die Exec-Library eröffnen wollen, schreiben Sie:

```
LIBRARY "exec.library"
```

Amiga Basic sucht die Libraries immer im aktuellen Verzeichnis. Notfalls müssen Sie mit dem Befehl CHDIR die Diskette oder das Unterverzeichnis, in denen sich die .bmap Dateien befinden, zum aktuellen Verzeichnis erklären. Nachdem die Library eröffnet wurde, steht uns nichts mehr im Wege, eine Systemroutine aufzurufen. Dazu dient der Basic-Befehl CALL. Nach dem Call-Befehl folgt der Name der Systemroutine und in Klammern die Parameter die diese Routine benötigt. Sie können den Befehl Call allerdings auch weglassen. Wenn Sie das tun, dürfen Sie die Parameter nicht in Klammern setzen.

Parameter

Noch ein Wort zu den Parametern. Diese müssen übrigens Variablen vom Typ Integer sein. Einige Systemroutinen geben uns einen Wert zurück. Diese Routinen müssen mit dem Befehl DECLARE FUNCTON definiert werden. Auch hierfür ein Beispiel:

```
DECLARE FUNCTION ReadPixel
&LIBRARY
```

Wie Sie sehen, muß hinter dem Namen der Routine das Zeichen für Long Integer stehen. Wenn die Library nicht mehr benötigt wird, sollten Sie diese wieder schließen. Dazu dient der Befehl LIBRARY CLOSE. Sie können übrigens 5 Libraries zur gleichen Zeit geöffnet haben. Wenn Sie des öfteren eine Library öffnen, ohne Sie vorher zu schließen, kann es passieren, daß der Amiga Ihnen die Guru Meditation-Meldung präsentiert. Diese unbeliebte Meldung sehen Sie auch sehr schnell, wenn Sie beim Aufruf der Systemroutinen unzulässige Parameterwerte übergeben. Also beim Experimentieren mit den Libraries immer daran denken: erst das Programm speichern, dann starten. Bevor wir zu den einzelnen Systemroutinen kommen, noch eine Begriffserklärung. Bei einigen Systemroutinen werden als Parameter keine Werte sondern ein Zeiger auf eine Struktur verlangt. Unter einer solchen Struktur können Sie sich einen Speicherbereich vorstellen, in dem wie in einer Tabelle verschiedene Daten stehen. Es können Werte von 1,2 oder 4 Byte Länge sein oder wiederum ein Zeiger auf eine andere Struktur. Solche Strukturen sind ein wichtiges Element beim Amiga. In C oder Assembler ist das Anlegen solcher Strukturen kein Problem. Amiga Basic tut sich da etwas schwerer. Aus diesem Grunde habe ich ein Programm geschrieben, welches das Anlegen einer solchen Struktur mit Amiga Basic vereinfacht. Dieses Programm macht uns gleich mit einer Betriebssystemroutine aus der Intuition Library bekannt.

Eine nützliche Hilfe

Diese Routine heißt AllocRemember und liefert beim Aufruf einen Wert zurück. Aus diesem Grunde muß diese Routine als Funktion definiert werden. Die Werte der anzulegenden Struktur werden in Datazeilen abgelegt. Als ersten Wert schreiben Sie b,w oder l. Der Buchstabe b bedeutet, daß der zu speichernde Wert 1 Byte lang ist, w steht für 2 Byte und l für einen 4 Byte langen Wert. Dahinter steht der eigentliche Wert. Soll ein Text in einer Struktur angelegt werden, so setzen Sie als Zeichen t und dahinter den Text. Wenn in der Struktur ein Zeiger auf eine weitere Struktur oder auf einen Text

```
' Programm Lib.Read
' Liest den Inhalt der System Librarys
'
CHDIR"df0:fd1.2"
INPUT"Name der Library :";ln$
'
ln$=ln$+"_lib.fd"
OPEN ln$ FOR INPUT AS #1
'
WHILE NOT EOF(1)
  LINE INPUT #1,buf$
  PRINT buf$
WEND
END
```

Listing 1: Lib. Read ließt die Systemlibraries

```
' Programm : Strukturmaker
'
DECLARE FUNCTION AllocRemember& LIBRARY
LIBRARY "intuition.library"

teststruktur:
DATA b,12
DATA b,0
DATA w,1200
DATA l,12991
DATA l,0     ' Zeiger auf Text
DATA t,hallo
DATA e

RESTORE teststruktur:GOSUB memory
RESTORE teststruktur:GOSUB struktur
POKEL pointer&+8,tpointer&(1)  ' Zeiger auf Text

GOSUB memoryclear    ' Speicher wieder freigeben
LIBRARY CLOSE
STOP

struktur:
```

Listing 2: Strukturmaker machts möglich.......

```
t=1:pointer&=speicher&
 READ byte$
 WHILE byte$<>"e"
   READ wert$
   IF byte$="b" THEN
     POKE speicher&,VAL(wert$)
     speicher&=speicher&+1
   END IF
   IF byte$="w" THEN
     POKEW speicher&,VAL(wert$)
     speicher&=speicher&+2
   END IF
   IF byte$="l" THEN
     POKEL speicher&,VAL(wert$)
     speicher&=speicher&+4
   END IF
   IF byte$="t" THEN
    tpointer&(t)=textsp&+offs(t)
     FOR i=1 TO LEN(wert$)
      POKE textsp&+offs(t)+i-1,ASC(MID$(wert$,i,1))
     NEXT
    POKE textsp&+offs(t)+i-1,0
     t=t+1
   END IF
   READ byte$
 WEND
RETURN

memory:
 t=1:maxsp&=0:textsp&=0
 READ byte$
 WHILE byte$<>"e"
   READ wert$
  IF byte$="b"THEN maxsp&=maxsp&+1
  IF byte$="w"THEN maxsp&=maxsp&+2
  IF byte$="l"THEN maxsp&=maxsp&+4
   IF byte$="t" THEN
     offs(t)=textsp&
    textsp&=textsp&+LEN(wert$)+1
     t=t+1
   END IF
```

vorkommt, so setzen Sie in den Datazeilen den Wert 0 ein, wenn Sie die Adresse dieser Struktur noch nicht wissen. Nach dem Aufruf der Routine Struktur poken Sie den Wert in die entsprechende Speicherstelle (siehe Beispiel im Listing). Nun wird mit RESTORE der Zeiger auf die Datazeilen gesetzt und die Routine memory aufgerufen. In dieser Routine wird der Speicherbedarf ermittelt und der Speicher mit der Systemroutine AllocRemember angelegt. AllocRemember hat folgendes Format:

speicher&= AllocRemember-(rememberkey,size,flags)

- rememberkey ist ein Zeiger auf eine Struktur, in der Intuition sich merkt, welchen Speicher Sie mit AllocRemember belegt haben. Setzen Sie diesen Zeiger einfach auf 0. - size definiert die Größe des benötigten Speichers.
- flags definiert folgendes:

- 2^0 = PUBLIC
- 2^1 = CHIP MEMORY
- 2^2 = FAST MEMORY
- 2^{16}= Speicherinhalt löschen

Die Routine AllocRemember hat den Vorteil, daß der belegte Speicher mit der Routine FreeRemember wieder dem System zurückgegeben werden kann. Selbst wenn Sie die Routine AllocRemember öfter aufrufen, brauchen Sie sich nicht darum zu kümmern, welche Speicherbereiche Sie belegt haben. AllocRemember merkt sie sich.
FreeRemember hat folgendes Format:

Call FreeRemember (remember key,reallyforget)

- rememberkey ist oben erwähnter Zeiger.
- reallyforget hat entweder den Wert 0, wenn nur die Rememberstruktur gelöscht werden soll, oder 1 wenn der Speicher und die Rememberstruktur für das System wieder verfügbar sein sollen.

```
    READ byte$
  WEND

  flag&=2^1+2^16
  speicher&=AllocRemember& (0,maxsp&,flag&)
  IF speicher&=0 THEN ERROR 7
  IF textsp&=0 THEN skip
  textsp&=AllocRemember& (0,textsp&,flag&)
  IF textsp&=0 THEN ERROR 7
  skip:
RETURN

memoryclear:
  CALL FreeRemember (0,1)
RETURN

```

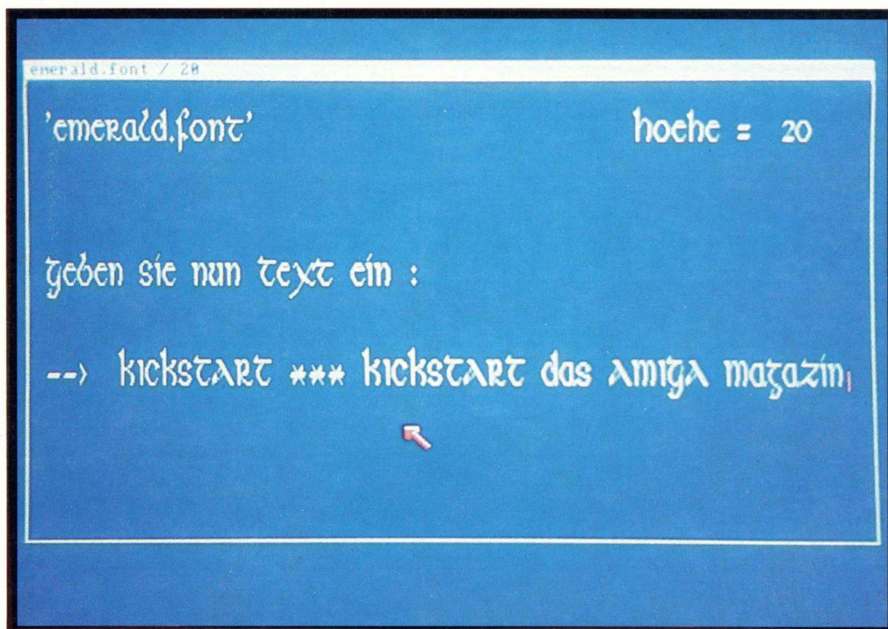
Die Zukunft

Ich hoffe, daß Sie nun das nötige Grundwissen haben, um mit mir die einzelnen Systemroutinen besprechen zu können. In der nächsten Folge werden wir uns mit der Graphics Library beschäftigen. Für Basic-Programmierer sicher ein Leckerbissen, dem Amiga ein wenig in die Karten zu schauen.

VON ERNST HEINZ

SCHRIFT WECHSEL

Systemfonts in Amiga BASIC



Auf dem Amiga wird die gesamte Monitorausgabe ausschließlich mittels Pixelorientierter Darstellungen in verschiedenen Grafikmodi erzeugt. Eine Trennung in rein textorientierte und spezielle Grafikausgabemodi, wie sie Ihnen sicherlich von anderen, älteren Computermodellen bekannt sein dürfte, gibt es beim Amiga somit nicht mehr.
Wie man an die Schriftarten in Basic herankommt, zeigt dieser Artikel.

Text wird innerhalb von pixelorientierten Darstellungen am einfachsten und effizientesten über sog. Punktmatrizen erzeugt, wobei natürlich für jedes auszugebende Textzeichen (Buchstabe, Ziffer oder Sonderzeichen) eine eigene, ganz spezielle Punktmatrix festgelegt werden muß.
Eine bis maximal 256 solcher Punktmatrizen mit aufeinanderfolgendem ASCII-Codewert der von ihnen dargestellten ASCII-Zeichen können unter dem Amiga-Grafiksystem zu einem ganzen Zeichensatz zusammengeschlossen werden. Dabei werden diese Zeichensätze - im Fachjargon auch Fonts genannt - mit einem Namen benannt und auf Diskette oder im freien RAM bzw. Pseudo-ROM Ihres Amiga abgespeichert. Der Vorteil dieser Art der pixelorientierten Textausgabe mit Hilfe von Punktmatrizen besteht zweifelsohne darin, daß das Erscheinungsbild und die Größe der Zeichen vom Benutzer oder Programmierer beliebig festgelegt werden können. Allerdings ist es aufgrund programmtechnischer Gründe nicht ratsam, die Größe der Punktmatrix innerhalb eines einzigen Zeichensatzes variabel zu gestalten. Das Amiga-Grafiksystem unterstützt deshalb auch nur Zeichensätze von konstanter Höhe - in Bezug auf die Breite der einzelnen Ausgabezeichen eines Zeichensatzes besteht dieser Zwang jedoch nicht.

Systemfonts

Das als kleine, theoretische Einführung zu Zeichensätzen auf dem Amiga.

Damit nicht jeder nun erst einmal seinen eigenen Zeichensatz mit 256 verschiedenen Punktmatrizen erstellen muß, bevor er überhaupt Text auf dem Monitor seines Amiga ausgeben kann, werden auf der Workbench-Systemdiskette in dem "fonts"-Verzeichnis schon einige vorgefertigte Systemfonts mitgeliefert.

Neben dieser Sammlung von bereits einsatzbereiten Fonts werden noch zwei weitere, fertige Fonts bereits während der Installation des Amiga-Betriebssystems in das Pseudo-ROM geladen: "topaz/8" und "topaz/9". Je nach Einstellung des gewählten Ausgabebetriebsmodus (80-Z oder 60-Z Darstellung), wird einer dieser beiden Fonts nach der erfolgreichen Systeminstallation erst einmal voreinstellungsgemäß aktiviert und für alle zu erledigenden Textausgaben verwendet.

Darüber hinaus erzeugt das Betriebssystem während der Systeminstallation ein logisches Gerät "FONTS:", unter dem die Systemfontsammlung immer zu finden ist. Nach dem Systemstart mit einer Workbench-Diskette wird diesem logischen Gerät "FONTS:" z.B. der Pfadname "SYS:fonts" zugewiesen sein.

1. Einzelheiten zum Umgang mit Zeichensätzen

Um einen neuen Zeichensatz für die Textausgabe verwenden zu können, muß man natürlich den zugehörigen Textfont, der ja die einzelnen Zeichenmatrizen zur Verfügung stellt, erst einmal aktivieren. Dieser Aktivierungsprozeß gliedert sich unter dem Amiga-Grafiksystem in zwei verschiedene Schritte:

(a) Textfont für die Benutzung öffnen, wobei auf Diskette gespeicherte Fonts evtl. erst noch in den Speicher geladen werden müssen.

(b) Textfont endgültig für die Tex tausgabe setzen und einem Ausgabe-RastPort zuordnen.

Zuerst muß der entsprechende Textfont also geöffnet werden.

Grundsätzlich unterteilt man Textfonts bezüglich ihres Speicherstatus in zwei Typen: DiskFonts (auf Diskette gespeichert) und MemoryFonts (im freien Speicher, also RAM oder Pseudo-ROM, abgelegt).

In Abhängigkeit vom Speicherstatus des zu öffnenden Textfonts (DiskFont oder MemoryFont) können diese über Aufrufe der folgenden Systemfunktionen geöffnet werden:

```
Font& = OpenFont&(TextAttr&)
für MemoryFonts

Font&=OpenDiskFont&(TextAttr&)
für DiskFonts
```

Beide Systemfunktionen liefern als Funktionsergebnis einen Adreßzeiger auf einen sog. TEXTFONT-Datensatz, dessen genaue Struktur uns im Rahmen dieses Artikels jedoch nicht näher interessieren soll.

Dieser TEXTFONT-Adreßzeiger wird in Schritt (b) benötigt. Ein Funktionsergebnis von NIL = not in list (Zahlenwert Null) besagt dabei, daß der angegebene Textfont nicht geöffnet werden konnte.

Doch zurück zu den beiden Systemfunktionen, die in zwei verschiedenen Systembibliotheken zu finden sind, nämlich in "OpenFont&" in der "graphics.library" und in "OpenDiskFont&" in der "diskfont.library". Beide erwarten als Parameter einen Adreßzeiger auf einen TEXTATTR-Datensatz, der die wichtigsten Eigenheiten wie Namen oder Höhe des zu öffnenden Textfonts charakterisiert.

Den genauen Aufbau eines TEXTATTR-Datensatzes im Detail zeigt Ihnen die Abbildung Nr.1, die diesen Datensatz in Form eines Pascalschen RECORD-Verbunds aufschlüsselt - in seine einzelnen Komponenten zerlegt! Hinter jeder RECORD-Komponente können Sie dann darüberhinaus noch den für Sie als AmigaBASIC-Programmierer äußerst wichtigen, relativen Byte-Offset derselben sowie deren Datengröße erfahren (B=Byte, W=WORD, L=LONG).

Die Bedeutung des eben angesprochenen Byte-Offsets wird Ihnen spätestens bei der weiter unten abgedruckten Besprechung des Beispielprogramms "FontLister" vollkommen bewußt werden.

Doch nun zur Erklärung der Bedeutungen der einzelnen Komponenten eines TEXTATTR-Datensatzes:

```
ta_Name
```

Ein Adreßzeiger auf eine mit einem NULL-Zeichen = CHR$(0) terminierte Zeichenkette, die den reinen Namen des Textfonts (ohne weitere Höhenbezeichnung !) angibt, also z.B. "topaz.font" oder "ruby.font".

```
t a_YSize
```

Eine positive Ganzzahl, welche die Höhe der Zeichen des Textfonts in Pixeln angibt.

```
ta_Style
```

Eine 2-Byte Marke, deren einzelne Bits den für diesen Textfont festgesetzten Zeichentyp charakterisieren (zur Bedeutung der einzelnen Zahlenwerte s.a. Abbildung Nr.1).

```
ta_Flags
```

Eine 2-Byte Marke, deren einzelne Bits voreingestellte Status- und sonstige Typwerte dieses Textfonts beschreiben (zur Bedeutung der einzelnen Zahlenwerte s.a. unten in Abbildung Nr.1).

Öffnen

Im Falle des TEXTATTR-Datensatzes, den Sie den Systemfunktionen "OpenFont&" oder "OpenDiskFont&" übergeben, beziehen sich die einzelnen Angaben natürlich auf den gewünschten, zu öffnenden Textfont.

Die beiden Systemroutinen suchen dann unter dem logischen Gerät "FONTS:" den Textfont mit dem angegebenen Namen heraus, dessen Eigenschaften am besten zu den im TEXTATTR-Datensatz spezifizierten passen. Findet sich kein solcher Textfont, liefern sie, wie bereits erwähnt, als Funktionsergebnis den Wert NIL.

Aktivieren

Im Falle des erfolgreichen Öffnens eines Textfonts muß nun noch Schritt (b) durchgeführt werden, um diesen neuen Zeichensatz für die Textausgabe end-

```
(***************************************)
(*                                     *)
(*   Datenstruktur TEXTATTR            *)
(*          8 Bytes                    *)
(*   definiert von TEXT.h              *)
(*                                     *)
(***************************************)



TextAttr = RECORD

    ta_Name  : POINTER TO String;     (* Offset = 0  /  L  *)
    ta_YSize : WORD;                  (* Offset = 4  /  W  *)
    ta_Style : BYTE;                  (* Offset = 6  /  B  *)
    ta_Flags : BYTE                   (* Offset = 7  /  B  *)

END;



(*************************************************************)
(*                                                           *)
(*                                                           *)
(*  Folgende Werte sind für 'TextAttr.tf_Style' definiert :  *)
(*                                                           *)
(*   $00 = 0     FS_NORMAL        (normale Textzeichen)      *)
(*   $01 = 1     FSB_UNDERLINED   (unterstrichene Textzeichen) *)
(*   $02 = 2     FSB_BOLD         (fette Textzeichen)        *)
(*   $04 = 4     FSB_ITALIC       (kursive Textzeichen)      *)
(*   $08 = 8     FSB_EXTENDED     (gedehnte Textzeichen)     *)
(*                                                           *)
(*                                                           *)
(*************************************************************)
```

Abb. Nr. 1: Der Aufbau eines TEXTATTR-Datensatzes

gültig zu aktivieren.
Auch hierzu stellt das Amiga-Grafiksystem eine leicht zu handhabende Systemroutine zur Verfügung:

```
SetFont(Font&,RastPort&)
```

Die Systemprozedur "SetFont" aus der Systembibliothek "graphics.library" ordnet einem beliebigen Rastport einen beliebigen Zeichensatz zu. Der Parameter "Font&" entspricht hierbei einem Adreßzeiger auf einen TEXTFONT-Datensatz, den man z.B. bei einem Aufruf der Systemfunktionen "OpenFont&" oder "OpenDiskFont&" als Funktionsergebnis erhält.
Der Parameter "RastPort&" dagegen muß ein Adreßzeiger auf einen RASTPORT-Datensatz (der allgemeinen Steuerdatenstruktur für Ausgabeelemente des Amiga-Grafiksystems) sein. Jeder Bildschirm und jedes Fenster besitzen auf dem Amiga z.B. ihren eigenen RASTPORT-Datensatz.
Unter AmigaBASIC kann der Adreßzeiger auf den RASTPORT-Datensatz des momentan aktiven Ausgabefensters folgendermaßen bestimmt werden:

```
WindowRastPort& = WINDOW(8)
```

In diesem Zusammenhang sollten Sie vorallem die wichtige Feinheit beachten, daß ein Aufruf von "SetFont" immer nur den Ausgabezeichensatz eines einzigen RASTPORT-Datensatzes neu zuordnet. Da jedes Fenster, wie oben bereits erwähnt, seinen eigenen RASTPORT-Datensatz besitzt, bedeutet dies in der Praxis, daß neue Zeichensätze mittels "SetFont" ganz gezielt nur einem einzigen Fenster zugeordnet werden können - die Ausgabezeichensätze aller übrigen Fenster bleiben unverändert!

Schließen

Schon der gesunde Menschenverstand gebietet, alles wieder zu schließen, was man einmal geöffnet hat. Und ganz genauso verhält es sich auch mit Zeichensätzen unter dem Amiga-Grafiksystem. Hat man erst einmal einen Textfont mittels "OpenFont&" oder "OpenDiskFont&" geöffnet, sollte man ihn auch vor Ende des Programms, bzw. sobald er nicht mehr gebraucht wird, wieder schließen - vorallem um die von diesem Textfont in Anspruch genommenen Systemresourcen wieder freizugeben.
Hierzu stellt die Systembibliothek "graphics.library" dem Programmierer folgende Systemroutine zur Verfügung:

```
CloseFont(Font&)
```

Diese Systemprozedur erwartet als Parameter einen Adreßzeiger auf den TEXTFONT-Datensatz des zu schließenden Zeichensatzes.
Nach dem Aufruf von "CloseFont" existiert der angegebene TEXTFONT-Datensatz nicht mehr, und auch alle anderen mit ihm vorher verknüpften Speicherbereiche sind wieder freigegeben worden.

Diskfonts

Gerade bei der Arbeit mit auf Diskette abgespeicherten Textfonts kann andauerndes Nachladen vom externen Speichermedium oftmals sehr unangenehm sein, ja sogar störende Auswirkungen haben. Aus diesem Grund stellt das Amiga-Grafiksystem dem Programmierer eine Systemliste zur Aufnahme von geöffneten Textfonts zur Verfügung.
Die Aufnahme eines Textfonts in diese Systemfontliste bewirkt, daß dieser Textfont nur ein einziges Mal von Diskette geladen werden muß und hernach allen Benutzern bzw. Programmen uneingeschränkt zur Verfügung steht. Wird dann ein Textfont der Systemfontliste wirklich auf keinen Fall mehr gebraucht, kann er unter automatischer Freigabe aller von ihm benötigten Systemresourcen wieder aus dieser Liste entfernt werden.
Die Systembibliothek "graphics.library" stellt dem Programmierer zur Arbeit mit der Systemfontliste zwei Routinen zur Verfügung - die Prozedur "AddFont" sowie die Funktion "RemFont%".

```
AddFont(Font&)

Fehler% = RemFont%(Font&)
```

Beide erwarten als Parameter einen Adreßzeiger auf den TEXTFONT-Datensatz des hinzuzufügenden bzw. zu entfernenden Zeichensatzes. Existiert dieser Zeichensatz in der Systemfontliste überhaupt nicht, oder kann er aus irgendeinem sonstigen Grund nicht aus der Systemliste entfernt werden, liefert ein Aufruf von "RemFont%" als Funktionsergebnis einen Fehlerkode ungleich Null - ein Funktionsergebnis von Null bedeutet dementsprechend, daß die gewünschte Entfernungsaktion mit Erfolg ausgeführt werden konnte.

Fontfolter

Zum Abschluß dieses Abschnitts sei jetzt noch eine weitere Funktion aus der Systembibliothek "graphics.-library" erwähnt, die im Zusammenhang mit der Handhabung von Textfonts von Bedeutung ist:

```
AskFont(RastPort&,TextAttr&)
```

Diese Systemprozedur bestimmt die Charakteristika des momentan aktiven Ausgabezeichensatzes eines beliebigen RastPorts.
Der Parameter "RastPort&" entspricht dabei dem Adreßzeiger auf den gewünschten RastPort, und "TextAttr&" entspricht einem Adreßzeiger auf einen TEXTATTR-Datensatz, in den die herausgefundenen Eigenschaftswerte des aktiven Textfonts dann als Ergebnis hineingeschrieben werden.
Mit Hilfe des auf diesem Weg erhaltenen TEXTATTR-Datensatzes kann der momentan aktive Textfont z.B. jederzeit wieder nachgeladen oder aber auch einem anderen RastPort neu zugewiesen werden.

2. Bestimmung der TEXTATTR-Datensätze aller verfügbaren Textfonts

Unter Anwendung der im ersten Abschnitt erläuterten Grundlagen der Handhabung von Textfonts sind Sie inzwischen schon in der Lage, jedem beliebigen Fenster einen eigenen Ausgabezeichensatz zuzuordnen. Allerdings können Sie bisher bei der Initialisierung der TEXTATTR-Datensätze zur Charakterisierung der zu öffnenden Textfonts nur hoffen, daß der von Ihnen gewünschte auch wirklich in der Systemfontsammlung unter dem logischen Gerät "FONTS:" verfügbar ist - sicher sein können Sie sich bisher jedoch nicht.
Zum anderen besteht durchaus auch die Möglichkeit, daß in der Systemfontsammlung ein vollkommen neuer, Ihnen daher unbekannter Textfont vorhanden ist - diesen werden Sie also überhaupt nicht benutzen können.
Kurz und gut, was Sie in diesem Zusammenhang noch brauchen, ist ein Verfahren bzw. eine Routine, mit deren Hilfe Sie die Eigenschaften (am besten natürlich direkt die zugehörigen TEXTATTR-Datensätze) aller in der Systemfontsammlung verfügbaren Textfonts bestimmen können.

Die Suche nach den Fonts

Aber nur keine Angst, daß es sich um eine komplexe Lösung dieser Aufgabe handelt, sie ist vielmehr erstaunlich einfach, da die Systemprogrammierer des Amiga-Grafiksystems die Notwendigkeit einer solchen Routine schon vorausgesehen und diese entsprechend implementiert haben.

```
FehlBytes&= AvailFonts&
( Puffer&,PufferGroesse&,Typ%)
```

Diese Funktion wird Ihnen von der Systembibliothek "graphics.library" zur Verfügung gestellt. Sie durchkämmt die gesamte Systemfontsammlung nach einsetzbaren Textfonts und legt eine Infoliste über diese im Speicher an.
Dabei weist der Parameter "Puffer&" als Adreßzeiger auf einen bereits für die Benutzung reservierten Speicherbereich, in dem diese Infoliste dann abgespeichert werden kann. Die Größe dieses Arbeitsspeicherbereichs in Bytes muß der Routine über den Parameter "PufferGroesse&" übergeben werden.
Der Wert des Parameters "Typ%" legt schließlich fest, ob in der externen Systemfontsammlung nach DiskFonts (Typ%=AFFDISK%=2), oder aber in der internen Systemfontliste nach MemoryFonts (Typ%=AFFMEMORY%=1) gesucht werden soll.
Als Funktionsergebnis liefert diese Funktion die Anzahl von Bytes, um die der zur Verfügung gestellte Speicherbereich zu klein war, um die gesamte Infoliste aufnehmen zu können.
Ein Funktionswert von Null bedeutet demnach, daß alles reibungslos geklappt hat. Andernfalls muß der Arbeits-speicherbereich eben einfach ent-sprechend vergrößert werden, so daß er am Ende genau (PufferGroes&+Fehl Bytes&) Bytes umfaßt!

Dekodierung

Bleibt nun noch die Aufgabe, die so erhaltene Infoliste auch richtig zu interpretieren. Doch dazu muß man erst einmal ihren Aufbau im Detail kennen, weshalb Ihnen Abbildung Nr.2 diesen vor Augen führt.

Die von "AvailFonts&" im Pufferspeicherbereich angelegte Infoliste ist also nichts anderes als ein AVAILFONTSHEADER-Datensatz.

Dieser besteht, wie Abbildung Nr.2 zeigt, aus nur zwei Komponenten, die folgende Bedeutung haben:

afh_NumEntries

Eine Ganzzahl, deren Wert die Anzahl der gefundenen Textfonts angibt.

afh_AF

Ein Feld von AVAILFONTS-Datensätzen, wobei die Anzahl der Feldelemente dem Wert von "AvailFontsHeader.afh_NumEntries" entspricht.

Natürlich ist hier das Feld von AVAILFONTS-Datensätzen von besonderem Interesse für den Programmierer, da jedes einzelne Element dieses Felds die Charakteristika eines gefundenen, also auf jeden Fall verfügbaren Zeichensatzes enthält, wie Abbildung Nr.3 zeigt, welche die Struktur eines AVAILFONTS-Datensatzes im einzelnen wiedergibt.

Die (wiederum nur) zwei Komponenten eines solchen AVAILFONTS-Datensatzes haben folgende Bedeutung:

af_Type

Eine 2-Byte-Marke, die dem Typ des zugehörigen Textfonts (MemoryFont oder DiskFont) entspricht.

af_Attr

Ein ganzer TEXTATTR-Datensatz (kein Adreßzeiger !), dessen einzelne Komponenten mit den gefundenen Werten des zugehörigen Textfonts belegt sind.

Die letztere Komponente stellt somit tatsächlich das dar, was wir am Anfang dieses Abschnitts gesucht haben - einen mit gültigen Werten gefüllten TEXTATTR-Datensatz eines verfügbaren Textfonts.

Und da man über einen Aufruf der Systemfunktion "AvailFonts&" ja eine Liste aller innerhalb des Systems verfügbaren Textfonts erhält, sind somit auf diesem Weg auch die TEXTATTR-Datensätze aller innerhalb des Systems verfügbaren Textfonts zugänglich. Das bedeutet aber wiederum, daß alle innerhalb des Systems verfügbaren Textfonts geöffnet und einem RastPort zugeordnet werden können. Das Problem ist demnach tatsächlich vollständig und, wie man annehmen darf, zur vollsten Zufriedenheit aller Anwender gelöst.

3. Das nützliche AmigaBASIC-Programm "FontLister" als Beispiel

In den beiden obigen Abschnitten haben Sie in der Theorie eigentlich alles gelernt, was man im Zusammenhang mit der Handhabung und Auswahl von Textfonts wissen muß. Der tatsächlich praktische Bezug soll nun in diesem letzten Abschnitt anhand eines doch schon recht anspruchsvollen Beispielprogramms hergestellt werden. Hierbei den goldenen Mittelweg zwischen Nutzen, Leistungsfähigkeit, Komplexität und Aussagekraft des zu

```
(***************************************************************)
(*                                                             *)
(*                                                             *)
(* Folgende Werte sind für 'TextFont.tf_Flags' definiert :     *)
(*                                                             *)
(* $01=  1  FPB_ROMFONT        (Font ist aus ROM geladen)      *)
(* $02=  2  FPB_DISKFONT       (Font ist von Diskette geladen) *)
(* $04=  4  FPB_REVPATH        (von rechts nach links)         *)
(* $08=  8  FPB_TALLDOT        (Font für HIRES/NON-INTERL. Modus)*)
(* $10= 16  FPB_WIDEDOT        (Font für LORES/INTERLACED Modus) *)
(* $20= 32  FPB_PROPORTIONAL (Font mit variabler Zeichenbreite)*)
(* $40= 64  FPB_DESIGNED       (Fontgröße nicht alg. konstruiert!*)
(* $80=128  FPB_REMOVED        (Font ist nicht aktiv)          *)
(*                                                             *)
(*                                                             *)
(***************************************************************)


(***************************************************************)
(*                                                             *)
(*   Datenstruktur AVAILFONTSHEADER                            *)
(*              ?                                              *)
(*   definiert von DISKFONT.h                                  *)
(*                                                             *)
(***************************************************************)


AvailFontsHeader = RECORD

     afh_NumEntries : INTEGER;             (* Offset = 0 / L *)
     afh_AF         : ARRAY OF AvailFonts (* Offset = 2 / - *)
END;
```

Abb.. Nr.2: Der Aufbau eines AVAILFONTSHEADER-Datenatzes

```
(***********************************)
(*                                 *)
(*   Datenstruktur AVAILFONTS      *)
(*        10 Bytes                 *)
(*   definiert von DISKFONT.h      *)
(*                                 *)
(***********************************)



AvailFonts = RECORD

     af_Type : WORD;              (* Offset = 0  /  W  *)
     af_Attr : TextAttr           (* Offset = 2  /  -  *)

END;



(*****************************************************************)
(*                                                               *)
(*                                                               *)
(*  Für 'AvailFonts.af_Type' sind folgende Werte definiert :     *)
(*                                                               *)
(*  $0001=1  AFF_MEMORY (zugehöriger Textfont im Pseudo-ROM)     *)
(*  $0002=2  AFF_DISK   (zugehöriger Textfont auf Diskette)      *)
(*                                                               *)
(*                                                               *)
(*****************************************************************)
```

Abb. Nr.3: Der Aufbau eines AVAILFONTS-Datensatzes

präsentierenden Beispielprogramms zu finden, ist keineswegs einfach.
Hoffentlich entspricht das hier vorgestellte AmigaBASIC-Programm zumindest in dieser Hinsicht Ihren Anforderungen und Erwartungen.

Der Fontlister

Zunächst also zum Nutzen und der Leistungsfähigkeit des Programms.
Mit Hilfe von "FontLister" können Sie die Namen und Eigenschaften aller innerhalb des Systems im Speicher oder auf Diskette verfügbaren Textfonts herausfinden.
Weiterhin erlaubt Ihnen der "FontLister", für jeden gefundenen und somit verfügbaren Textfont eine Demo-Textausgabe in einem speziell dafür neu angelegten Ausgabefenster durchzuführen. Auf diese Weise können Sie sich mit dem "FontLister" das Erscheinungsbild aller Zeichen eines jeden verfügbaren Textfonts ganz in Ruhe anschauen. Da die gesammte Bearbeitung der Textfonts von Systemroutinen erledigt wird, erweist sich das Programm als effizient und schnell.
Neben den beiden, schon oben häufig angesprochenen, Systembibliotheken "graphics.library" und "diskfont.library" greift der "FontLister" zusätzlich noch auf die Systembibliothek "exec.library" zu. Aus dieser Systembibliothek werden nämlich die beiden Routinen "AllocMem&" und "FreeMem" zur Reservierung und Freigabe des Arbeitsspeicherbereichs für die Systemprozedur "AvailFonts" benötigt.

Her mit dem freien Speicher

Für diejenigen Leser, denen die Arbeitsweise dieser beiden Systemroutinen zur Speicherverwaltung noch nicht bekannt sein sollte, sei sie hier noch einmal erklärt, da sie in der Tat von fundamentaler Bedeutung ist.

```
MemBlock&=
AllocMem& ( BlockGroesse&,Art&)
FreeMem(MemBlock
&,BlockGroesse&)
```

Mittels der Funktion "AllocMem&" kann ein Speicherbereich der durch den Parameter "BlockGroesse&" angegebenen Größe (in Bytes) zur Benutzung reserviert, d.h. bereitgestellt werden. Dieser Bereich ist dann innerhalb des Systems nicht mehr frei verfügbar, und auf ihn kann nur noch direkt, also nicht mehr durch ein nachfolgendes "AllocMem&", zugegriffen werden. In welcher Art von freiem Speicher (Chip-RAM, Public-RAM usw.) er reserviert werden soll, wird durch die einzelnen Bits des Parameters "Art&", einer 4-Byte-Maske bestimmt. Ein Wert von $2^{16}+2^{0}=65537$ entspricht dabei z.B. Public-RAM, wobei der neu belegte Speicherbereich direkt auch noch automatisch mit lauter Nullen initialisiert wird. Die genauen Bedeutungen der einzelnen Bits dieser 4-Byte Maske entnehmen Sie bitte [2].
Als Funktionsergebnis liefert diese Funktion einen Adreßzeiger auf das erste Byte des neu reservierten Speicherbereichs. Ein Ergebnis von NIL (Zahlenwert gleich Null) bedeutet dabei, daß die gewünschte Reservierung nicht erfolgreich war (meist aus Mangel an freiem Speicherplatz).
Die Prozedur "FreeMem" gibt nun einen mittels "AllocMem&" erzeugten Speicherbereich wieder für das gesammte System frei. Der Parameter "MemBlock&" entspricht dabei dem bei einem erfolgreichen Aufruf von "AllocMem&" erhaltenen Ergebnisadreßzeiger auf das erste Byte des Speicherbereichs. Der Wert des Parameters "BlockGroesse&" gibt wiederrum die Größe des freizugebenden Speicherbereichs in Bytes an.

Weiter geht's

Doch nun zum strukturellen Aufbau von "FontLister".
Der Hauptprogrammteil beginnt mit der Initialisierungsphase, in der neben einigen wichtigen Pseudo-Konstanten wie z.B. AFFMEMORY% und AFFDISK% auch die benötigten Systembibliotheken geöffnet und die

verwendeten Systemfunktionen vom Ergebnistyp her deklariert werden.

Nach erfolgter Eingabe des Suchmodus (<D>isk oder <M>emory) durch den Benutzer erfolgt die eigentliche Bestimmung der verfügbaren Textfonts mittels der Systemprozedur "AvailFonts" innerhalb einer WHILE-Schleife. Diese bietet den Vorteil, evtl. notwendige Größenkorrekturen des an "AvailFonts" übergebenen Pufferspeicherbereichs ohne großen, weiteren Bedingungs /Abfrageaufwand bearbeiten zu können.

Nach erfolgreicher Bestimmung aller innerhalb des Systems verfügbaren Textfonts wird eine Liste derselben im aktuellen Programmausgabefenster angezeigt. Dies erledigt das Unterprogramm "AvailFontsListe" unter ständigem Aufruf der SUB-Unterroutine "listeFontInfo" für jeden AVAILFONTS-Datensatz der durch die Systemprozedur "AvailFonts" im "Puffer&" erzeugten Fontliste.

Hierbei wird gleichzeitig in "FontName$()" ein Feld der Namen aller verfügbaren Textfonts angelegt, um auf diese später sofort zugreifen zu können.

Aus der nun angezeigten Liste von Textfonts kann der Benutzer einen beliebigen Textfont für eine Demo-Textausgabe auswählen. Hierzu wird in dem Unterprogramm "PrintDemo" zunächst einmal der zu dem ausgewählten Textfont gehörige TEXTATTR-Datensatz aus der Liste von AVAILFONTS-Datensätzen herausgesucht und der Speichertyp (MemoryFont oder DiskFont ?) desselben bestimmt.

In Abhängigkeit vom Speichertyp kann der Textfont anschließend über einen Aufruf von "OpenFont&" oder "OpenDiskFont&" zur Benutzung geöffnet werden. Nach erfolgter Öffnung wird schließlich ein neues Fenster für die Demo-Textausgabe eröffnet, dem direkt danach der ausgewählte Textfont über einen Aufruf der Systemroutine "SetFont" als aktiver Ausgabezeichensatz zugewiesen wird.

Hat der Benutzer die Demo-Textausgabe dann beendet, werden das Demo-Ausgabefenster und der diesem zugeordnete Ausgabezeichensatz wieder geschlossen. Zum Schließen des Textfonts bedient man sich hierbei logischerweise der Systemprozedur "CloseFont" - fertig.

Zum Abschluß noch ein Tip für Besitzer eines Amiga-Minimalsystems mit nur einem Laufwerk.

Um einen ständigen Diskettenwechsel während der Ausführung des Programms zu vermeiden, kopieren Sie das Programm nebst zugehöriger ".bmap"-Dateien auf die RAM-Disk. Danach legen Sie die Systemstartdiskette mit der Systemfontsammlung wieder in Ihr eines Laufwerk.

So konfiguriert wird die Arbeit mit dem "FontLister" auch für Sie zu einem reinen Vergnügen!

Literatur

[1] "Amiga ROM Kernel Reference Manual: Libraries and Devices" Commodore Business Machines, Inc. Addison-Wesley

[2] "Amiga ROM Kernel Reference Manual: Exec" Commodore Business Machines, Inc. Addison-Wesley

[3] "Commodore AMIGA : Amiga-Basic" Commodore


```
'--------------------------------------------'
' Unterroutine "listeFontInfo" gibt anhand der AVAILFONTS-Struktur Infodaten
' zu beliebigen Fonts aus
' von Ernst Heinz / 29.06.1987
'--------------------------------------------'
'
' Bedeutungs des Parameters :
'
' AvailFontsZeiger&   - ein Adresszeiger auf AVAILFONTS-Daten
'  FontName$          - enthaelt am Ende den Namen dieses Fonts
'--------------------------------------------'

SUB listeFontInfo ( AvailFontsZeiger& , FontName$ ) STATIC

   SHARED  AFFMEMORY%                         ' globale Pseudo-Konstante
  FontNameZeiger&=PEEKL(AvailFontsZeiger&+2)   ' Adresszeiger auf Fontnamen
   Nm$=""                                     ' Fontnamen initialisieren
  PRINT " '";
  WHILE PEEK(FontNameZeiger&)<>0              ' Null-terminierter C-String
    Nm$=Nm$+CHR$(PEEK(FontNameZeiger&))        ' naechstes Zeichen ausgeben
    FontNameZeiger&=FontNameZeiger&+1         ' Zeiger inkrementieren
   WEND

  PRINT Nm$;"'";SPACE$(35-LEN(Nm$));           ' Name ausgeben
   FontName$=Nm$                              ' Namen des Fonts eintragen
  Nm$=STR$(PEEKW(AvailFontsZeiger&+6))         ' Fonthoehe bestimmen
  PRINT "Hoehe = ";Nm$;SPACE$(7-LEN(Nm$));     ' Fonthoehe bestimmen
  FontName$=FontName$+" /"+Nm$                ' Fonthoehe an Namen anhaengen
  AFTyp%=PEEKW(AvailFontsZeiger&)              ' Typwert des Fonts lesen
  IF (AFTyp% AND AFFMEMORY%)<>0 THEN          ' ist es ein MemoryFont ?
    PRINT "    MemoryFont";
   ELSE                                       ' oder ein DiskFont ?
```

```
    PRINT "    DiskFont";
  END IF
END SUB


'------------------------------------------'
'
' Demoprogramm "FontLister" gibt Informationen ueber die momentan verfueg-
' baren Systemfonts aus, die dem logischen Geraet FONTS: zugeordnet sind
' von Ernst Heinz / 29.06.1987
'
'------------------------------------------'
'
'
' Liste der verwendeten Systembibliotheksroutinen :
'
'  "diskfont.library"  -   FUNCTION  AvailFonts&
'                          FUNCTION  OpenDiskFont&
'
'  "graphics.library"  -   PROCEDURE SetFont
'                          PROCEDURE CloseFont
'                          FUNCTION  OpenFont&
'
'  "exec.library"      -   PROCEDURE FreeMem
'                          FUNCTION  AllocMem&
'
'------------------------------------------'


ON BREAK GOSUB CtrlCgedrueckt    ' BREAK-Unterbrechungen abfangen
BREAK ON

LIBRARY "diskfont.library"      ' zur Bearbeitung von DiskFonts
LIBRARY "graphics.library"      ' zur Bearbeitung von MemoryFonts
LIBRARY "exec.library"          ' zur Speicherverwaltung

DECLARE FUNCTION AvailFonts&   LIBRARY   ' Bestimmung der verfuegbaren Fonts
DECLARE FUNCTION OpenDiskFont& LIBRARY   ' zum Laden eines DiskFonts
DECLARE FUNCTION OpenFont&     LIBRARY   ' zum Laden eines MemoryFonts
DECLARE FUNCTION AllocMem&     LIBRARY   ' zur Speicherallozierung

AFFMEMORY% = 1                  ' Typwert fuer speicherresidenten Font
AFFDISK%   = 2                  ' Typwert fuer einen Diskettenfont
MEMFPUBLIC& = 1      ' = 2^0    ' Typwert fuer beliebigen RAM-Speicher
MEMFCLEAR& = 65536&  ' = 2^16   ' Typwert fuer Speicherinitialisierung
CLS : PRINT

PRINT "FontLister Version 1.0  (von Ernst Heinz / 29.06.1987)"
PRINT "----------------"
PRINT : PRINT

WHILE (Wahl$<>"M") AND (Wahl$<>"D")    ' solange bis korrekte Eingabe
  LINE INPUT "<D>iskFonts oder <M>emoryFonts listen ?  ",Wahl$
  Wahl$=LEFT$(UCASE$(Wahl$),1)        ' nur das erste Zeichen interessiert
WEND

PRINT : PRINT
PRINT "FontInfos werden gerade gelesen....."
PRINT : PRINT

IF Wahl$="M" THEN         ' Suchtyp fuer AvailFonts setzen
  Typ%=AFFMEMORY%         ' MemoryFonts listen
ELSE
  Typ%=AFFDISK%           ' DiskFonts listen
END IF

PufferGroesse&=302        ' Puffergroesse in Byte (reicht fuer 30 Fonts)
GOSUB erzeugePuffer       ' Datenpuffer im RAM anlegen
l&=1                      ' Schleifenbedingung mit TRUE initialisieren

WHILE l&<>0
  l&=AvailFonts&(Puffer&,PufferGroesse&,Typ%)   ' verfuegbare Fonts bestimmen
  IF l&<>0 THEN                                 ' ist der Puffer zu klein ?
```

```
    CALL FreeMem(Puffer&,PufferGroesse&)       ' ja -> Puffer loeschen !
    PufferGroesse&=PufferGroesse&+1&
     GOSUB erzeugePuffer                      ' neuen, groesseren Puffer !
  END IF
WEND
AvailAnz%=PEEKW(Puffer&)          ' Anzahl der verfuegbaren Fonts bestimmen
DIM FontName$(AvailAnz%-1)        ' ein Feld fuer alle FontNamen
WHILE Wahl$<>"0"                  ' Hauptprogrammschleife
  CLS
  GOSUB AvailFontsListe           ' Liste aller verfuegbaren Fonts ausgeben
  PRINT : PRINT
  LINE INPUT "( Ende = 0 )  PRINT-Demo fuer Font Nr.";Wahl$
  FontNr%=VAL(Wahl$)-1            ' Nr. des gewaehlten Fonts berechnen
  IF (FontNr%>=0) AND (FontNr%<=AvailAnz%-1) THEN  ' ist diese Nr. gueltig ?
     GOSUB PrintDemo                               ' ja -> Demoausgabe !!
  END IF
WEND
Schluss1:
CALL FreeMem(Puffer&,PufferGroesse&)  ' Pufferspeicher wieder freigeben
Schluss2:
PRINT  : PRINT
LIBRARY CLOSE                         ' Systembibliotheken wieder schliessen
END
'-------------------------------------'
'
' Unterprogramm "erzeugePuffer" alloziert einen Speicherbereich der gefor-
' derten "PufferGroesse&", auf welchen der Adresszeiger "Puffer&" deutet.
' Falls nicht genuegend Speicherplatz zur Verfuegung steht, wird das Pro-
' gramm einfach abgebrochen.
'
'-------------------------------------'
erzeugePuffer:
  Puffer&=AllocMem&(PufferGroesse&,MEMFPUBLIC%+MEMFCLEAR%)  ' Puffer erzeugen
     IF Puffer&=0 THEN                                     ' alles klar ?
     BEEP : PRINT : PRINT                                  ' nein !!
    PRINT "Nicht genuegend freier Speicherplatz vorhanden !"
     GOTO Schluss2                                         ' Abbruch !
  END IF
RETURN
'-------------------------------------'
'
' Unterprogramm "AvailFontsListe" gibt eine Liste aller verfuegbaren Fonts
' im Programmfenster aus
'
'-------------------------------------'
AvailFontsListe:
  FOR i%=0 TO AvailAnz%-1                    ' fuer jeden Font nun ein Info
    PRINT
    PRINT USING "###" ; i%+1 ;              ' jedem Font ein Nummer zuordnen
    PRINT ".  ";
   CALL listeFontInfo(Puffer&+2+i%*10,FontName$(i%))   ' FontInfo ausgeben
  NEXT i%
RETURN
'-------------------------------------'
```

```
'
' Unterprogramm "PrintDemo" eroeffnet ein neues Fenster und ordnet diesem
' dann den im Hauptprogramm ausgewaehlten Textfont zu.
' Danach wird in dem neuen Fenster ein kurzer Text ausgegeben.
'
'________________________________________'

PrintDemo:

  TextAttrZeiger&=Puffer&+2+FontNr%*10+2    ' Adresszeiger auf TEXTATTR-Daten
  FontTyp%=PEEKW(TextAttrZeiger&-2)         ' Typwert des Fonts bestimmen
  IF (FontTyp% AND AFFMEMORY%)<>0 THEN      ' ist es ein MemoryFont ?
    Font&=OpenFont&(TextAttrZeiger&)        ' mittels "OpenFont" laden
  ELSE                                      ' nein , es ist ein DiskFont !
    Font&=OpenDiskFont&(TextAttrZeiger&)    ' mittels "OpenDiskFont" laden
  END IF

  IF Font&=0 THEN                           ' Fehler beim Laden ?
    PRINT : PRINT : BEEP
    COLOR 3,1                               ' inverse Textausgabe
    PRINT "Dieser Font kann nicht geladen werden !";
    COLOR 1,0                               ' wieder normale Textausgabe
    WHILE INKEY$=""                         ' auf <RETURN> warten
    WEND
  ELSE                                      ' nein , alles o.k. !
    WINDOW 2,FontName$(FontNr%),,16         ' Demo-Fenster
    CALL SetFont(WINDOW(8),Font&)           ' Font aktivieren
    PRINT : PRINT " ";
    CALL listeFontInfo(Puffer&+2+10*FontNr%,s$) ' Demo-Textausgabe
    PRINT : PRINT : PRINT
    PRINT " Geben Sie nun Text ein :" : PRINT
    LINE INPUT " -> ";s$
    WINDOW CLOSE 2                          ' Demo-Fenster schliessen
    CALL CloseFont(Font&)                   ' und Font schliessen
  END IF


'________________________________________'
'
' Unterprogramm zur Bearbeitung von BREAK-Unterbrechungen
'
'________________________________________'


CtrlCgedrueckt:

  RETURN                        ' BREAK-Unterbrechungen sind wirkungslos


'________________________________________'

```

VON FLORIAN DU BOIS

KAVALIERS START

Der Bootblock

Sie haben sich sicherlich schon einmal gewundert, wie schnell manchmal ein Bild oder ein Farbenscrolling nach dem Einlegen einer Diskette erscheint. Nach dem Einschalten des Computers oder nach einem Reset verlangt der Computer eine Diskette (beim A1000 die Kickstartdiskette oder die Workbench-Diskette, beim A500/A2000 nur die Workbenchdiskette, weil das Kickstart bei Ihnen im ROM ist). Legt man anschließend eine Diskette ein, läuft das Laufwerk kurz an, geht wieder kurz aus und lädt dann weiter. Was lädt der Computer in dieser kurzen Zeit nach dem Einlegen der Diskette ? Was passiert in der Zeit, in der das Laufwerk kurz ausgeht?

Nach jedem Einlegen einer Diskette lädt der Computer immer als erstes den sogenannten Bootblock. Dieser Bootblock belegt die ersten beiden Blöcke jeder Diskette. Ist er auf einer Diskette nicht vorhanden, wird diese vom Computer nicht angenommen. Entweder kann diese Diskette nur von der Workbench aus benutzt werden oder sie ist nur eine Datendiskette eines Programms und muß deshalb nicht selbst starten können. Eine unformatierte Diskette nimmt er natürlich auch nicht an, weil auch dort kein Bootblock vorhanden ist.

Wie ist der Bootblock aufgebaut ?

Zuerst kommen 4 Bytes, die den Diskettentyp angeben. Es gibt 3 verschiedene Diskettentypen:

1.DOS-Diskette

Eine DOS-Diskette ist eine normale, ladbare Diskette (z.B.: Workbench). Diese wird durch die Buchstaben 'DOS' und ein folgendes Nullbyte gekennzeichnet.

2.Kickstart-Diskette

Die Kickstart-Diskette wird beim Einschalten des A1000 verlangt und durch die Buchstaben 'KICK' gekennzeichnet.

3.Sonstige Disketten

Sonstige Disketten sind
a) Datendisketten
b) Unformatierte oder andersformatierte Disketten

Bei diesen Disketten findet der Computer weder das Wort 'DOS' noch das Wort 'Kick' und nimmt sie nicht an.

Nach der Bezeichnung des Diskettentyps folgt ein Langwort, in dem die Checksumme des Bootblocks liegt. Diese Checksumme ist Summe aller im Bootblock befindlichen Worte. Der Computer errechnet Sie nach dem Laden des Bootblocks und vergleicht das Ergebnis mit dem Checksummenlangwort auf der Diskette. Stimmen die beiden Summen nicht überein, wird der Ladevorgang abgebrochen. Stimmen sie überein, wird das Programm gestartet, das mit dem 4. Langwort des Bootsektors beginnt.

Die Blocknummer des sogenannten Root-Blocks (In diesem Block sind der Diskettenname, das Hauptdirectory und das Erstellungsdatum der Diskette gespeichert.) liegt im nächsten Langwort. Normalerweise ist dies der Block 880.

Nun folgt ein Programm, das jedesmal beim Einlegen einer Diskette geladen wird.

Was für ein Programm liegt im Bootblock ?

Normalerweise liegt hier ein Programm, welches überprüft, ob die Dos-Library vorhanden ist. Der Befehl 'Install' schreibt dieses kleine Programm in die beiden ersten Blöcke einer Diskette. Möchte man also eine formatierte Diskette für den Computer brauchbar machen, dann muß man nur den Install-Befehl aufrufen. Man kann mit diesem Befehl auch Programme überschreiben, die im Bootblock liegen, die aber nur stören (SCA-Virus siehe Kickstart 11/87).

Der 'normale' Bootblock sieht so aus (Sekaformat):

```
dc.b "DOS",0            ;Bezeichnung einer Dos-Diskette
blk.l 1                 ;Platzhalter für die Checksumme (ein Langwort)
dc.l 880                ;Blocknummer des Rootblocks

Programm:

lea Dosname(Pc),a1      ;Adresse des Namens der Dos-Library in A0
jsr -96(a6)             ;Sprung in die Findresident-Routine der Exec-Library
                        ;Routine testet ob die Doslibrary vorhanden ist
tst.l d0                ;Testen ob D0=0
beq negativ             ;wenn nicht, dann Sprung nach Negativ
move.l d0,a0            ;Wert aus D0 in A0
move.l 22(a0),a0        ;Zeiger auf die Initialisierungsroutine des Dos in a0
moveq #0,d0             ;D0 löschen

Ende:

                        ;Programm beenden und Sprung zur Initialisierungsrou-
                          tine

Negativ:
move #$ff,d0            ;-1 in D0 schreiben
bra Ende                ;und zum Ende springen

Dosname:
dc.b "dos.library"
```

Nachdem das Programm geprüft hat, ob die Dos-Library vorliegt, schreibt es in D0 eine -1, wenn sie nicht vorliegt, oder eine 0, wenn sie vorliegt. Außerdem schreibt es in A0 die Adresse der Initialisierungsroutine des Dos.

Wie legt man ein eigenes Programm in den Bootblock ?

Will man selbst ein Programm in den Bootblock legen, das direkt nach dem Einlegen der Diskette gestartet wird, muß man auf mehrere Dinge achten. Das von 'Install' installierte Programm muß auf jeden Fall im Bootblock bleiben. Das eigene Programm muß Pc-orientiert (Pc = Programmcounter) geschrieben werden, weil man nicht weiß, wo der Computer das Programm im Speicher ablegt. Bevor man sein eigenes Programm startet, muß man die Register A0 und D0 retten, weil in ihnen wichtige Zahlen gespeichert sind (siehe Install-Programm). Man muß auch bedenken, daß das Programm nicht sehr lang sein darf, da man nur zwei Sektoren für es hat. Weil nicht alle Befehle des Installprogramms unbedingt notwendig sind, kann man es kürzen und hat damit ein bißchen mehr Platz für sein eigenes Programm. Ein Beispiel eines Bootblockprogramms finden Sie im Anschluß an diesen Artikel. Sie müssen Ihr Programm dann mit 'wi' (beim Seka) abspeichern. Mit einem normalen Monitor können Sie dann das Programm in die Bootsektoren legen. In einem Monitor ist meistens auch eine Funktion zum Bestimmen der Checksumme vorhanden, durch die Sie dann die Checksumme für ihren Bootblock erhalten.

Wie nutzen professionelle Programme den Bootsektor ?

Wenn Sie 'Pinball Wizzard' von Kingsoft kennen, haben Sie sich wahrscheinlich gefragt, wie das Titelbild so schnell erscheinen kann. Sie wissen jetzt, daß ein Programm im Bootblock der Diskette liegen muß, welches das Erscheinen des Bildes bewirkt. Wenn Sie jetzt diesen Artikel gelesen haben, müßten Sie sich die Frage stellen, wie ein Programm und ein Bild in den kleinen Bootblock passen sollen. In diesem Fall werden die Bilddaten nachgeladen (für Programmierer : mit der DoI0-Routine aus der Exec-Library). Das Nachladen ist natürlich eine Möglichkeit, auch längere Programme vom Bootblock aus zu laden. Allerdings verlängert sich dadurch die Zeit des Ladens, und die normale Schnelligkeit des Bootblocks wird verringert. Eine weitere Möglichkeit, den Bootblock zu benutzen, besteht darin, in ihm ein Ladeprogramm zu installieren, welches das Programm der Diskette lädt. Dies ist zum Beispiel bei den Spielen Barbarien und Terrorpods von Psygnosis der Fall. Dadurch ist es auch möglich, Disketten mit einem anderen Format zu lesen.
Außerdem gibt es Programme, wie zum Beispiel den Bootboy von Atlantis, die es dem Benutzer erlauben, selbstgestaltete Programmvorspänne in den Bootblock zu legen. Bei diesen Programmen ist man aber sehr eingeschränkt, selbst etwas am Diskettenvorspann zu ändern. Beim Bootboy kann man zum Beispiel nur die Farben des scrollenden Hintergrunds, die Farben des herrunterscrollenden Objekts und das Objekt verändern. Sie sehen, wie wenig Möglichkeiten der eigenen Gestaltung man bei solchen Programmen hat. Anstatt sich ein solches zu kaufen, kaufen Sie sich lieber ein Monitorprogramm, mit dem Sie ihr eigenes Programm in den Bootblock legen können.

Ein Beispiel für ein Bootblockprogramm (Seka)

Legt man das folgende Programm in den Bootblock und startet die Diskette, wechselt der Bildschirm ständig seine Farbe. Dies ist ein sehr einfaches und sicherlich nicht sehr interessantes Programm. Es soll mit ihm aber auch nur das verkürzte Install-Programm deutlich gemacht und ein Beispiel für die 'Bootblockprogrammierung' gegeben werden. Wichtig hierbei ist, daß die Register gerettet werden, so daß sie nach der Beendigung des Programms unverändert dem Betriebssystem übergeben werden. Außerdem ist es wichtig, daß Sprünge innerhalb des Programms Pc-orientiert sind (jsr $xxxx(pc) oder jmp $xxxx(pc)). Ebenso muß man den Befehl 'lea adresse,An' zu 'lea adresse(pc),An' (Siehe Install-Programm : lea dosname(pc),a1) und den Move-Befehl 'move.l #Adresse,An' zu 'lea Adresse(pc),An umschreiben.

Doch sehen Sie sich jetzt einmal das Beispielprogramm an, an dem Sie die ganze Theorie vielleicht besser verstehen.

```
Start:
dc.b "DOS",0                  ; \
dc.l 0                        ;  \
dc.l $370                     ;   \
lea libname(pc),a1            ;    \  verkürztes
jsr -96(a6)                   ;    /  Install-Programm
move.l d0,a0                  ;   /
move.l 22(a0),a0              ;  /
clr.l d0                      ; /

Hautprogramm:

movem.l D0-D7/A0-A6,-(sp)     ;Retten der Register
move.l 4,A6                   ;execbase in A6
move.w #$03a0,$dff096         ;DMA's sperren
clr.l D0                     ;D0 löschen (Bilschirmfarbe=0)
loop:
addi.w #$1f,D0               ;Bildschirmfarbe erhöhen
move.w D0,$DFF180            ;Bildschirmfarbe in Farbregister 0 schreiben
move.l #$3,D3                 ; \
                              ;  \
loop3:                        ;   \
move.l #$ffffffff,D2          ;    \ Verzögerungsschleife,weil es sonst
                              ;   /
loop2:                        ;  /   flackert
dbra D2,loop2                 ; /
dbra D3,loop3                 ; /
btst #6,$BFE001              ;testen, ob die linke Maustaste gedrückt ist
bne loop                     ;wenn nicht, dann neue Farbe in d0
move.w #$83e0,$dff096        ;wenn ja,nötige DMAs freigeben movem.l (SP)+,D0-
D7/A0-A6                     ;gerettete Register zurück
rts                          ;und ins Betriebssystem zurückspringen

libname:
dc.b "dos.library"

Ende:
```

Das Programm ist mit wi (Seka-Assembler) abzuspeichern, wobei bei 'begin' Start und bei 'end' Ende eingegeben werden muß (vor dem Abspeichern muß das Progamm assembliert worden sein). Dieses mit wi abgespeicherte Programm muß jetzt mit einem Monitor-Programm direkt in den Bootblock gelegt werden. Ich benutze den C-Monitor von Diamond Software. Dieser Monitor ist sehr gut für unseren Zweck geeignet, weil man mit ihm ein Programm an eine beliebige Stelle im Speicher legen, es dann dort bearbeiten und anschließend in irgendwelche Sektoren legen kann. Außerdem ist auch eine Funktion zur Bestimmung der Bootblockchecksumme vorhanden. Hat man irgendeine Funktion des Monitors vergessen, bekommt man durch die Eingabe von 'h' + <Return> eine Liste aller Befehle und deren Handhabung ausgegeben. Doch nun zu unserem Bootblockprogramm.

Zuerst sucht man sich mit '[Bytezahl' einen freien Platz im Speicher, weil man sonst möglicherweise ein Programm (z.B.: Das Monitor-Programm) überschreiben würde, und es dann zum Absturz des Rechners käme. Als Bytezahl muß man mindestens 1024 ($400) angeben, weil man ja Platz für zwei Sektoren braucht. Der Computer gibt dann aus, wieviel Bytes er 'freimachen' konnte und an welcher Stelle im Speicher. Jetzt lädt man das mit dem Seka abgespeicherte Programm mit 'L Prgname Anfangsadresse des freien Speichers' in den Computer (Das 'L' muß großgeschrieben sein, weil es sonst etwas anderes bewirkt). Anschließend läßt man mit 'b Anfangsadresse' die Checksumme berechnen (Das 'b' muß klein sein). Das Programm muß jetzt nur noch auf die ersten beiden Sektoren der Diskette gebracht werden. Dies macht man mit dem Befehl '>s Anfangsadresse Track Head Sector'. Für Track gibt man 0 ein, denn der Bootblock befindet sich auf der Spur 0 einer Diskette. Ebenso gibt man bei Head 0 ein, weil sich der Bootblock auf der Seite 0 einer Diskette befindet. Für den Sektor muß man einmal 0 und das zweite Mal 1 eingeben. Speichert man den zweiten Teil des Programms im Sektor 1 ab, muß man daran denken die Anfangsadresse um 512 ($200) zu erhöhen, weil man die ersten 512 Bytes ja schon im Sektor 0 abgespeichert hat. Hat man das alles gemacht, befindet sich das Bootprogramm auf der Diskette. Wahrscheinlich hat sich das jetzt alles etwas kompliziert angehört, doch das ist es garnicht.

Ich zeige Ihnen jetzt noch einmal die Zeilen, die Sie eingeben müssen und die der Computer ausgibt, nachdem Sie das Monitorprogramm gestartet haben. Ich gehe davon aus, daß Sie das Programm mit dem Seka als 'bootprg' abgespeichert haben, und daß der Computer den Speicher ab $30000 für den Bootblock freihält.

Nachdem der Monitor gestartet wurde:

```
|400                                          ;$400 Bytes Speicher frei
Allocated 400 at $30000                       ;ab Adresse $30000 frei
L bootprg 30000                               ;bootprg nach $30000
Loading bootprg to $30000-$30073              ;bootprg wird geladen
b30000                                        ;Bootblockchecksumme
Oldchecksumme $00000000 corrected to $48c67b1 ;$48c67b1=neue
Checks.
>s30000 0 0 0                                 ;Sektor 0 beschreiben
Cyl: 0  Hd:0  Sec:0  Err:0  $30000-$30200     ;
>s30200 0 0 1                                 ;Sektor 1 beschreiben
Cyl: 0  Hd:0  Sec:0  Err:0  $30200-$30400     ;(+$200)
```

und schon ist das Bootprogramm auf der Diskette (Jede zweite Zeile gibt der Computer aus).
Die Checksumme gilt natürlich nur für das beschriebene Programm. Bis wohin das Progamm geladen wird, hängt von seiner Länge ab.
Sie sollten beachten, daß Sie sämtliche Zahlen und Adressen in der Hexadezimalschreibweise eingeben müssen, und daß Sie beim Schreiben des Programms auf die Diskette immer nur die ersten beiden Sektoren beschreiben, weil sonst möglicherweise wichtige Daten der Diskette gelöscht werden. Sie sollten auch darauf achten, daß Sie die Befehlsbuchstaben in der richtigen Größe schreiben. Haben Sie nur einen anderen Monitor zur Hand, dürfte es wahrscheinlich nicht schwer sein, ein Bootprogramm auch mit diesem in den Bootblock einer Diskette zu bringen.Ich wünsche Ihnen viel Spaß beim Schreiben eigener Bootprogramme.

DER GROßE KICKSTART

HARDWARE WETTBEWERB

1.PREIS 3000.-

2.PREIS 2000.-

3-10. PREIS: JE 20 PD DISKETTEN

WER KANN MITMACHEN?

ES IST JEDE PERSON AN UNSEREM HARDWAREWETTBEWERB BETEILIGT, DIE BIS ZUM EINSENDESCHLUß EINEN FERTIGEN PROTOTYP UND DIE SCHALTBESCHREIBUNG EINSENDET. DIE SCHALTUNG MUß NATÜRLICH FREI VON RECHTEN DRITTER UND DARF BIS JETZT NOCH NICHT IN EINERANDEREN ART VERÖFFENTLICHT WORDEN SEIN.

WAS DARF ES SEIN?

GEFRAGT SIND BEI DIESER AUSSCHREIBUNG ALLE ARTEN VON HARDWARE.

TEILNAHMEBEDINGUNGEN:

DIE VERGABE DER PREISE ERFOLGT DURCH EINE AUS REDAKTIONSMITGLIEDERN GEBILDETE JURY. DER RECHTSWEG IST AUSGESCHLOSSEN.

ALLE EINGESANDTEN SCHALTUNGEN, SCHALTPLÄNE UND PROTOTYPEN WERDEN DURCH DIE JURY AN DIE EINSENDER ZURÜCKGESCHICKT.

DER TEILNEHMER GIBT DURCH SEINE EINSENDUNG DIE ERKLÄRUNG AB, DAß DIE SCHALTUNG FREI VON RECHTEN DRITTER IST.

DAS COPYRIGHT DER PREISE1UND 2 GEHT AN DIE MERLIN COMPUTER GMBH.

MITARBEITERN DER MERLIN COMPUTER GMBH.

MITARBEITERN DER MERLIN COMPUTER GMBH UND DEREN ANGEHÖRIGEN IST DIE TEILNAHME UNTERSAGT.

SCHICKEN SIE IHRE SCHALTUNG MIT BESCHREIBUNG UND PROTOTYP AN FOLGENDE ADRESSE :

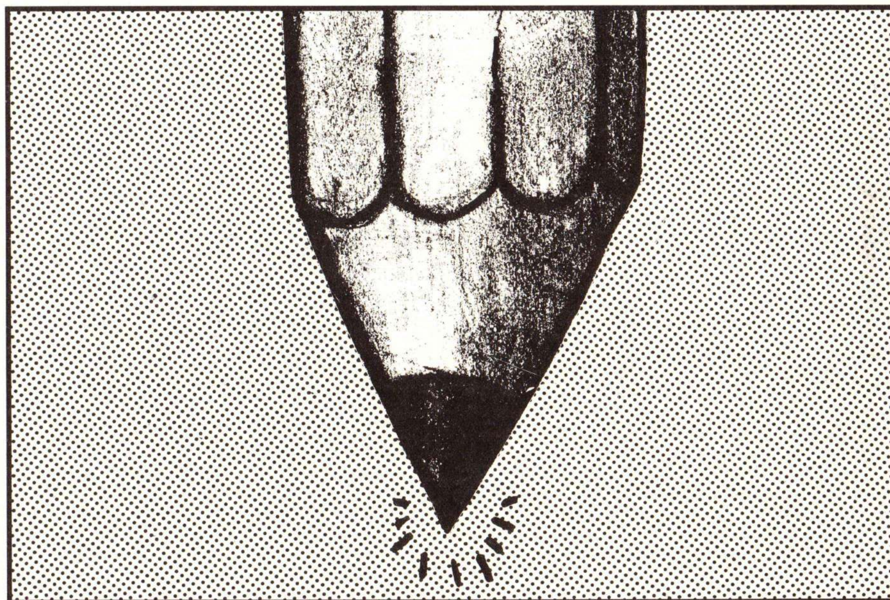
MERLIN COMPUTER GMBH, KENNWORT KICKSTART-HARDWAREWETTBEWERB, INDUSTRIESTRAßE 26, 6236 ESCHBORN

VON GERALD CARDA

ZAUBERN MIT DEM STAB

Ein Lichtgriffel am AMIGA

Der Computerbesitzer von heute schreit nach immer besseren und komfortableren Eingabegeräten. Als Apple vor wenigen Jahren die Maus als Eingabemedium einführte, hielten viele 'Experten' das für zu verspielt und kindisch. Heute denkt man auf diesem Gebiet ganz anders, obwohl der krampfhafte Versuch mancher Programmierer, Abläufe, die erwiesenermaßen schneller über die Tastatur zu erreichen wären, diese umständlich und zeitaufwendig über die Maus zu steuern, nicht gerade zu der Beliebtheit dieser Steuermöglichkeit beigetragen hat. Betrachtet man sich den heutigen Markt an harter und weicher Ware, stellt man fest, daß die Maus nicht mehr vom Tisch zu rollen ist.



Eine weitere, wenn nicht sogar direktere Eingabehilfe stellt der Lichtgriffel da. Dieses, in der englischsprechenden Welt (also auch hierzulande) Lightpen genannte, Eingabegerät ist noch etwas näher am Geschehen auf dem Bildschirm. Der Anwender muß nicht erst eine Maus oder einen Trackball in Bewegung setzen, sondern kann mit der Spitze des Lichtgriffels einfach auf die entsprechende Stelle auf dem Bildschirm zeigen und die Aktion wird ausgeführt.
Das hier zum Test vorliegende Exemplar der Firma Inkwell Systems nennt sich schlicht 'the LightPen'. Ob das Gerät seinem leicht suggestiven Namen gerecht wird, mußte sich erst zeigen.

Lieferumfang

Die Verpackung des LightPen enthält außer diesem selbst noch eine Diskette sowie ein Handbüchlein. Zum Handbuch selbst ist nicht viel zu sagen. Sicher haben Sie schon öfter einmal einen Joystick in Port 2 Ihres Rechners gesteckt; nun, genauso einfach ist der Anschluß des Lichtgriffels. Die Entwickler des Amiga haben ihrem Kind die Fähigkeit mitgegeben, ein solches Gerät über diesen Port abzufragen. Leider haben sie die nötige Software vergessen.
Hier kommt die mitgelieferte Diskette ins Spiel. Sie enthält neben dem benötigten Treiberprogramm auch noch die ausführliche Beschreibung der einzelnen Möglichkeiten dieser Software. Zu beachten ist hierbei, daß zwei unterschiedliche Treiberprogramme für die Modelle A1000 und A500/A2000 existieren. Diese Tatsache hat ihren Grund im unterschiedlichen Aufbau der Portbuchsen bei den einzelnen Modellreihen.

Praxistest

Der Lichtgriffel ist mit einer ausreichend langen Anschlußleitung versehen. Das etwa 13 Zentimeter lange Gehäuse des LigtPen hat eine dreieckige Form und liegt gut in der Hand. Am vorderen Ende ist, wie erwartet, eine Linse angebracht, die einen dahinterliegenden Fototransistor steuert. Die Linse ist ca. 8 Millimeter nach innen versetzt eingebaut. Dies verhindert zwar eine Beschädigung- zum Beispiel durch Zerkratzen der Linsenoberfläche-, macht aber gleichzeitig ein Reinigen des Strahlengangs zu einer umständlichen Prozedur. Ein etwas kleinerer Abstand hätte hier auch genügt, zumal sich durch die statische Aufladung an der Bildschirmoberfläche schnell Staub und Schmutzpartikel anlagern.
Ist die Treibersoftware geladen, kann man die Maus vergessen. Der Treiber des LightPen hat Vorrang vor dem systemeigenen Treiber der Maus. Nach entsprechender Einstellung des Treiberprogramms folgte der Zeiger brav den Bewegungen des LightPens. Zwei am vorderen Ende angebrachte Tasten ersetzen die Maustasten. Aus Platzgründen wählten die Konstrukteure hier keine mechanischen Tasten sondern sensorische. Leider haben die meisten Menschen nicht die nun nötige Fingerfertigkeit. Die kleinste, unbeabsichtigte Berührung löst schon die entsprechende Aktion aus.

DPaint

Die interessanteste Anwendung eines Lichtgriffels ist sicher die in einem Zeichenprogramm. Sie können quasi auf dem Bildschirm malen und zeichnen, eine allemal direktere Methode als über die Maus. Leider kommt bei solchen Anwendungen die Position des Bildschirms stark zum Tragen. Der in kurzem Abstand senkrecht aufgestellte Bildschirm vermittelt ein Schreibgefühl ähnlich dem liegend von innen an die Badewannenwand. Eine stärkere Bildschirmneigung und eine andere Sitzposition (das Optimum ist in diesem Fall ein in den Schreibtisch eingelassener Monitor) könnten hier helfen, wenn sich nicht noch einige andere Mängel gezeigt hätten.

Ein Mangel

Ein Lichtgriffel reagiert auf den Zeichenstrahl der Bildröhre und meldet so die momentane Position auf dem Bildschirm. In diesem Prinzip ist ein grundsätzlicher Schwachpunkt des Verfahrens begründet. Es arbeitet nicht auf schwarzen bzw. sehr dunklen Flächen. Diesen Mangel kann man noch durch das Wählen einer etwas helleren Hintergrundfarbe, wie zum Beispiel Dunkelgrau, umgehen.

Eine Kinderkrankheit

Der zweite Mangel, der bei dem Test mit DPaint aufgetreten ist, ist da schon um einiges folgenschwerer. Die Treibersoftware will es in der jetzigen Version nicht schaffen, den Stift dem LightPen folgen zu lassen. Beim Freihandzeichnen bzw. -Schreiben verliert der auf dem Bildschirm folgende Zeiger ca. einen Zentimeter auf fünf Zentimeter zurückgelegten Weg. Dieser Fehler ist eindeutig durch die Treibersoftware begründet.

Fazit

Der mit ca. 285,- DM nicht gerade billige LightPen ist mit der vorliegenden Treibersoftware nur eine Spielerei und kann für ernsthafte Anwendungen nicht empfohlen werden. Der Hauptvorteil eines Lichtgriffels kommt mit dieser Treibersoftware nicht zum Tragen. Man kann nur hoffen, daß der Hersteller sein Versprechen einlöst und so schnell wie möglich eine verbesserte Version der Software in den Handel bringt. Gerade für Mal- und Zeichenbegeisterte bzw. Personen des grafischen Gewerbes ist ein funktionierender Lichtgriffel in Verbindung mit einem leistungsfähigen Zeichenprogramm eine unheimliche Arbeitserleichterung.

Hersteller:Inkwell Systems
Kalifornien
Anbieter: IM Frankfurt
Tel.: 069/7071102

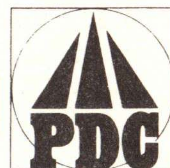
IHR AMIGA-VERSTAND SAGT: NIMM DEN PDC-VERSAND!



CRAZY CARS	64,90
TEST DRIVE	74,95
BACKLASH	54,90
OBLITERATER	64,90
MERCENARY	64,90
TIME BANDIT	54,90
XENON	54,90
ROADWARS	54,90
JUMP JET	44,95
FLIGHT SIMULATOR II	99,00
WINTER OLYMPICS	54,90
INSAMTY FIGHT	75,00
MOUSE TRAP	44,95
Q-BALL	59,00
SHADOWGATE	89,00
INTO THE EAGLES NEST	75,00
SPACE RANGER	29,95
FEUD	29,95
DARK CASTLE	75,00
PLUTOS	49,95
IMPACT	49,95
GALILEO	105,00
GIRLS OF RIVIERA	44,95
TERRORPODS	74,95
INT-SWITCH	27,50

ART OF CHESS	64,90
SARGON III	95,00
CHESSMASTER 2000	80,00
ANIMATE 3D	275,00
PIXMATE	120,00
BUTCHER 2.0 deutsch	110,00
DIGIPIC	1100,00
AEGIS IMPACT	150,00
AEGIS VIDEO TITLER	249,00
MIDI GOLD	170,00
THE DIRECTOR	125,00
PRISM +	120,00
PROJECT D	85,00
DYNAMIC DRUMS	135,00
DYNAMIC STUDIO	375,00
MAXIPLAN 500	249,00
MAXIPLAN PLUS	345,00
PROFESSIONAL PAGE	660,00
TURBO-PRINT	69,00
AEGIS SONIX	135,00
MEGACOVER (PVC, A500 + Maus)	29,90
MOUSE MAT	16,50
MOUSE MOUSE	19,90
ROBO CITY NEWS	4,95

DER VERSAND-PARTNER VON



GTI GmbH

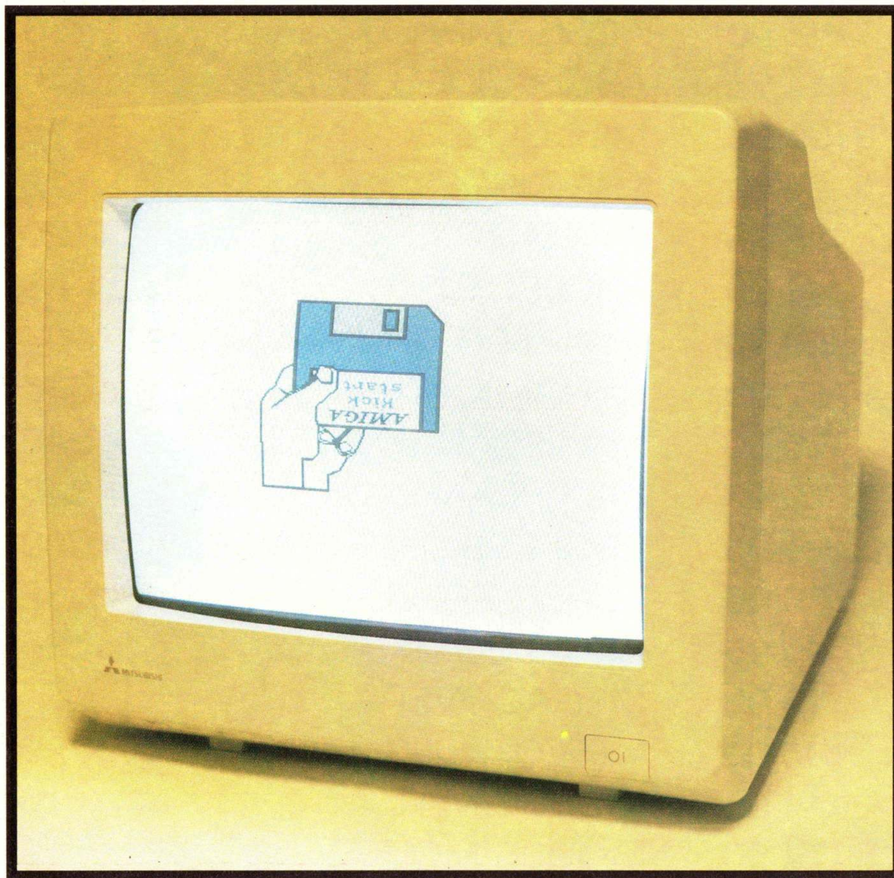
OBERHÖCHSTADTER STR. 53 B · 6370 OBERURSEL · TEL. 0 61 71 / 5 38 63

NACHNAHME 6,- DM · VORKASSE 4,- DM · AUSLAND: NUR GEGEN VORKASSE 10,- DM

VON GERALD CARDA

DER KONTAKT MANN

Mitsubishi-Monitor EUM-1471A



Viele Amiga-Anwender sind mit der Qualität des Commodore-Monitors nicht zufrieden. Der Markt bietet hier, für einen entsprechend größeren Betrag, bessere Monitore an. Der Bildausgang des Amiga verlangt einen RGB-Analog-Monitor, um die Bilddaten korrekt wiedergeben zu können. Fast alle Monitore mit dieser Fähigkeit gehören der Gattung der Multisync- bzw. Multiscan-Monitore an.

Auch der Mitsubishi EUM-1471A gehört zu dieser Montiorart. Es können Bilder mit einer Horizontalfrequenz von 15,6 bis 35 kHz dargestellt werden. Die Bildwiederholfrequenz kann hierbei zwischen 45 und 75 Hz liegen. Der Monitor ist somit auch für zukünftige Signalquellen gut gerüstet. Der Test des Mega-Vision Grafiksubsystems zeigt eine der möglichen Verwendungsarten.

Außenansicht

Das Gerät paßt in seiner Farbe gut zum Amiga. In der Standardlieferung ist leider kein Monitorständer enthalten. Besitzer eines Amiga 500 kommen so nicht umhin, einen Monitorständer nachzukaufen. Anwender eines Amiga 1000 oder 2000 können den Monitor, wie das Original, auf den Rechner stellen. Allerdings biegt sich das Kunststoffgehäuse des Amiga 1000 hierbei schon etwas durch. Der Mitsubishi EUM-1471A ist mit knapp 13 kg nicht gerade ein Leichtgewicht.

An Einstellmöglichkeiten befinden sich auf der Vorderseite außer dem Netzschalter die Regler für Kontrast und Helligkeit. Der Farbregler ist auf der Geräterückseite untergebracht, wird jedoch bei Analogbetrieb nicht verwendet, so daß dies keine Einschränkung des Bedienungskomforts darstellt.

Desweiteren finden sich folgende Regler auf der Rückseite: H-POSI, V-POSI, H-SIZE, V-SIZE, SCAN MODE, Eingangswahlschalter und ein Monochromschalter. Mit diesen Reglern ist die Bildposition auf dem Monitor, unabhängig vom Rechner, zu

zentrieren. Diese Prozedur ist nur einmal nötig und nach ca. zwei Minuten erledigt. Der Schalter SCAN MODE erlaubt die Darstellungsarten Over- und Underscan. Das Bild kann somit in horizontaler Richtung bis an den Bildröhrenrand gestreckt werden. Der Eingangswahlschalter wählt die Betriebsart des Monitors. Es stehen drei Modi zur Auswahl: eine Analogbetriebsart, ein TTL- und ein Videomodus. In der TTL-Betriebsart ist der Monochromschalter wirksam. Dieser Schalter ermöglicht es, den Monitor über eine Monochromkarte am PC zu betreiben.

Kontaktfreudig

Der Monitor bietet für jede mögliche Betriebsart einen separaten Eingang. Es können so ohne lästiges Umstecken drei Signalquellen am Monitor angeschlossen sein, deren Auswahl dann über den Betriebsartenwahlschalter erfolgt, der über drei Leitungen am Analogeingang ferngesteuert werden kann. Der Monitor ist so zum Beispiel, ohne Kabelumstecken, in folgender Konfiguration betriebsfähig: Amiga 2000 über Analogeingang, XT-Karte mit AGA-Grafikkarte über TTL und ein Fernsehempfänger über VIDEO. Die Auswahl des dargestellten Bildes wird einfach über den Betriebsartenschalter vorgenommen.
Leider ist dem Tatendrang des frischgebackenen Monitorbesitzers eine Schranke gesetzt. Der Lieferumfang schließt zwar eine kurze, aber gute Bedienungsanleitung und ein Netzkabel ein, enthält aber als Signalkabel nur den 9-poligen PC-Adapter. Sie können jedoch auch ein Kabel nach der abgedruckten Skizze selbst anfertigen. Ein fertiges Kabel kann jedoch auch bei der unten aufgeführten Adresse bezogen werden.

Aktion

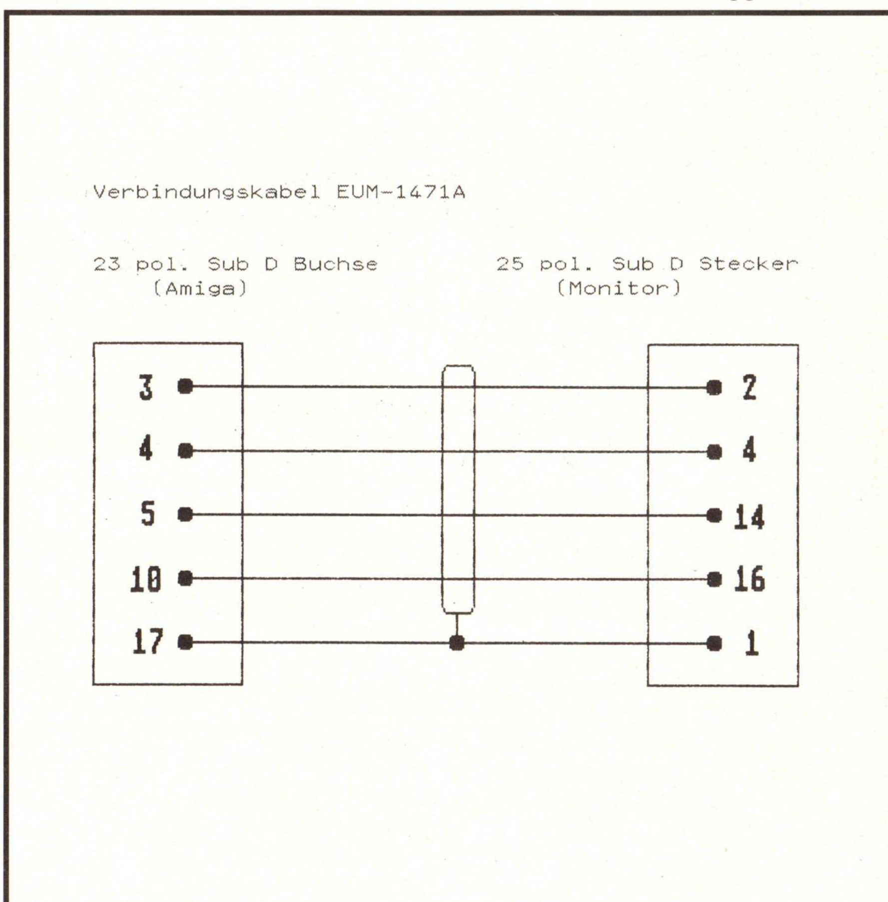
Nach der Anfertigung eines entsprechenden Kabels konnten wir zum erstenmal das Bild begutachten. Der Monitor liefert durch einen Punktabstand von 0,31 Millimeter ein deutlich schärferes Bild als der Commodore 1084. Die Bildröhre verfügt über einen guten Kontrast und ist entspiegelt. Die Bildschirmvergütung wird von Mitsubishi als Diamant-Matt bezeichnet.
Unser Testbild zeigt Ihnen die Möglichkeiten des Monitors. Zu beachten ist eine leichte Neigung des Bildes nach rechts. Diese Art von Bildfehler läßt sich jedoch in einer Fachwerkstatt leicht beheben und stellt in unserem Fall nur einen Ausreißer dar.

Interlace

In dieser Betriebsart haben viele Monitore ihre Schwierigkeiten. Der Mitsubishi EUM-1471A liefert in diesem Darstellungsverfahren ein annehmbares Bild, obgleich er die Qualität des NEC Multisync in dieser Disziplin nicht ganz erreicht. Anwender, die häufiger im Interlacemodus arbeiten müssen (z.B. CAD), sollten auf ein langnachleuchtendes Modell zurückgreifen, daß vom selben Hersteller angeboten wird. Ein solches Gerät testen wir gesondert in unserem Grafiksonderheft.

Fazit

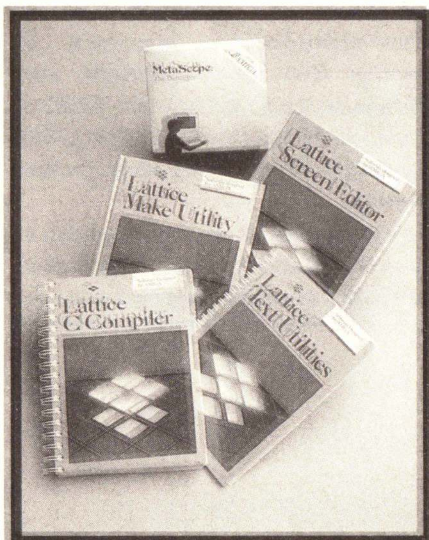
Der Mitsubishi EUM-1471A stellt einen leistungsfähigen Monitor dar. Besonders für Besitzer eines Amiga mit PC-Teil ist dieser Monitor ein vielseitig einsetzbares Gerät. Durch den getrennten Videoeingang ist es auch für Besitzer von Videorekordern oder Fernsehempfängern interessant. Es verfügt über die Möglichkeit, zwei Eingangssignale (Video und RGB) zu mischen. Diese Feature konnte jedoch innerhalb dieses Tests nicht probiert werden.
Etwas störend ist der im Vergleich zu anderen Monitoren etwas laute Zeilentrafo, der deutliche Zirpgeräusche von sich gibt. Beim Betrieb an einem lüfterlosen Amiga 500 stört dieses Geräusch deutlich. Der Amiga 2000 übertönt es jedoch mit seinem nicht gerade leisen Lüfter. Wenn Sie dieser Geräuschpegel nicht weiter stört, können Sie jedoch auch problemlos mit dem Zirpen leben.



Preis: ca. 1399.-
Hersteller: Mitsubishi
Vertrieb:
EAS Computertechnik
4630 Bochum 1
Tel: 0234/34307

VON MATHIAS KEGELMANN

LATTICE C 4.0



Nachdem Manx vor einigen Monaten seine neueste Version Aztec 3.4 (siehe KICKSTART 9/87) veröffentlichte, liegt nun auch von Latti-ce eine neue Version des bekannten Compilers vor. Der Umfang der Änderungen seit dem Lattice 3.1 (siehe Kickstart 1/87) veranlaßte Lattice sogar dazu, die Versionsnummer auf 4.0 zu erhöhen.

Erste Eindrücke

Von den 4 Compilerdisketten sind, wie in der vorhergehenden Version, nur zwei für das alltägliche Arbeiten wichtig. Die anderen beiden enthalten Beispielprogramme, Dokumentation, einige sehr nützliche Public Domain Tools und ähnliches. Von den beiden Hauptdisketten ist eine bootbar, d.h. man kann sie nach dem Anschalten (beim A1000 nach dem Einlesen der Kickstart) oder nach einem Reset anstatt der Workbench-Diskette einlegen. Diese Diskette enthält u.a., wie eine Workbench, sämtliche CLI-Befehle. Zusätzlich sind im Befehlsdirectory noch die zum Compiler gehörenden Kommandos, d.h. die beiden Compilerteile, den Compilertreiber, der Linker, ein Disassembler, ein Librarian, mit dem man sich eigene Funktionsbibliotheken erstellen kann und zusätzlich ein Macroassembler, der jetzt zum Amigastandard kompatibel ist. Dazu jedoch später mehr. Auf der zweiten Diskette befinden sich die Libraries, die beim Linken eingebunden werden und die normalen Amiga-Headerfiles, die jedoch in komprimierter Form vorliegen, was die Geschwindigkeit des Compilers erhöht. Die unkomprimierten, dokumentierten Headerfiles findet man auf Disk 3, was ein recht guter Kompromiß ist, da man so jederzeit die dokumentierten Files zur Hand hat. Der Compiler arbeitet aber nicht wie früher mit Files, in denen einfach die Kommentare gelöscht wurden, sondern mit noch weiter komprimierten und somit für den Compiler günstigeren arbeitet.Ferner findet man auf dieser Diskette noch die Sources einiger kleiner Routinen, die beim Linken normalerweise eingebunden werden, um beispielsweise die Kommandozeile auszuwerten. Angenehm überrascht ist man, daß Lattice in der normalen Startup-Sequence gleich zwei sehr nützliche Public Domain Programme aufruft. Das ist zum einen die neueste Version des PopCLI, der es ermöglicht, jederzeit durch gleichzeitiges Drücken von AMIGA+ESC einen neuen CLI zu eröffnen, was bei einem Software-Error ohne Guru häufig der letzte Weg ist, wichtige Files noch von der Ramdisk auf eine "echte" Diskette zu retten. Ferner schaltet der PopCLI nach einer bestimmten Zeit (Default ist 5 min), während der keine Taste gedrückt und die Maus nicht bewegt wurde, den Bildschirm ab, um die Bildröhre zu schonen. Sobald man dann wieder eine beliebige Taste drückt oder auch nur die Maus bewegt, wird der Bildschirm wieder angezeigt. Des weiteren wird noch ein bisher weitgehend unbekanntes Tool, der CONMan gestartet, der einen neuen Console-Handler einbindet, der volles Edieren der Eingabezeile ermöglicht und zusätzlich eine History-Funktion beinhaltet, d.h. vorher getippte Zeilen lassen sich durch einfaches Drücken von Cursor hoch wieder aufrufen und dann edieren. Das erspart viel Ärger, wenn man sich in einer langen Zeile irgendwo in der Mitte vertippt hat.Dies wirkt auf jedes neu geöffnete CON:-Fenster und macht somit aus jedem neuen CLI eine Mini-Shell, bei der man jedoch, da es keine echte Shell ist, sondern sie eine Ebene tiefer in Systemroutinen eingreift, keine neuen Befehle und Regeln lernen muß, da die normalen CLI-Befehle aufgerufen werden, als wäre der CONMan garnicht vorhanden. Dieses Tool wird man, besonders wenn man wegen eines kleinen Speichers eine Shell nicht vernünftig betreiben kann, nicht mehr missen wollen.Danach ruft die Startup-Sequence eins von zwei Batchfiles auf, das die vom Compiler benötigten Parameter entweder für normale Diskette oder Harddisk einstellt.Hierbei tauchen die ersten Probleme auf, wenn man nicht mindestens zwei Laufwerke besitzt. Die von Lattice empfohlene Mindestkonfiguration ist 512k und 2 Laufwerke. Wenn möglich, sollte man jedoch über mehr RAM, da man dann zeitaufwendige Diskzugriffe durch die intensive Benutzung der Ramdisk umgehen kann, oder eine Harddisk verfügen. Man kann den Compiler zwar auch mit nur einem Laufwerk betreiben, muß dann aber entweder die CLI-Kommandos ins RAM kopieren, hat somit nur noch wenig Platz für das eigentliche Programm, das man schreiben will, oder man muß andauerndes Diskwechseln in Kauf nehmen.Der Amiga meldet sich dann durch den CLI-Prompt, da der gesamte Compiler vom CLI aus benutzt wird. Die Programmentwicklung von einer graphischen Benutzeroberfläche ist nämlich nicht sinnvoll, sie wäre denkbar umständlich. Braucht man dennoch eine Workbench, um beispielsweise die "Verträglichkeit" eines Programmes mit dieser zu testen, kann man diese jederzeit durch 'LOADWB' aktivieren.

Besondere Optionen des Compilers

Nun zum Compiler selbst. Er besteht, wie sämtliche Vorgängerversionen, aus zwei Teilen: LC1 und LC2, die den beiden Compilerpasses gleichkommen. Diese können zwar getrennt aufgerufen werden, was aber so gut wie nie sinnvoll ist, da man sich damit nur unnötig Arbeit schafft. Normalerweise ruft man deshalb den sogenannten Compilertreiber LC auf, der sämtliche Optionen der beiden Passes kennt, LC1 und LC2 jeweils mit den entsprechenden Parametern aufruft und schließ-

lich, wenn man die -L Option angibt, auch das entstandene Programm mit den entsprechenden Startup-Programmen und den nötigen Libraries linkt. So kann man nicht nur einzelne Programme übersetzen und linken, sondern, da man auch Wildcards benutzen darf, ist es auch möglich, mehrere Programme zu kompilieren und dann alle zusammenzulinken.

LC -L ab?

übersetzt dann z.B. alle Sourcefiles, die mit ab beginnen und danach ein beliebiges Zeichen enthalten. Z.B.: ab1.c, ab2.c, ab3.c, abm.c.Dies ist eine Minimalimplementierung eines Make-Befehls, die für einfachere Anwendungen - beispielsweise wenn alle Sourcefiles mit den gleichen Options kompiliert werden sollen - durchaus ausreicht. Ein komplettes Make Utility ist aber, wie bereits anfangs erwähnt, als getrenntes Programm, wie das üblich ist, im Professional Package enthalten. Dazu jedoch später mehr.Der Compiler unterstützt eine weitere Option, die einige Mühe und Tipparbeit erspart: Will man nämlich eine Funktion in eine eigene Library einbinden, ist es nicht nötig, den Librarian getrennt aufzurufen. Man kann einfach die Option -R und danach den Namen der Library angeben. Dadurch wird nach der Kompilation die Funktion automatisch in die Library aufgenommen und eine ältere Funktion gleichen Namens gegebenenfalls gelöscht.

16 Bit INTs

Einer der größten Unterschiede zwischen Lattice und Aztec C war bisher, daß der Lattice Compiler INTSs 32 Bit-Breite, also Longwords, übersetzt, wie dies in Kernighan&Ritchie für 68000 Rechner empfohlen wird. Der Aztec-Compiler dagegen benutzt die schnelleren und kürzeren 16 Bit-INTs. Ferner ärgerte es Lattice-Programmierer hin und wieder (wenn sie sich den Kode ansahen, den der Compiler erzeugt hatte), daß gleich, was als Parameter für einen Funktionsaufruf auf den Stack gelegt wurde, immer zunächst auf 32 Bit erweitert wurde. Der Aztec dagegen legt für Bytes und Words immer 16 Bit auf den Stack, da der Stackpointer sowieso niemals ungerade werden kann. Nur bei wirklich 32 Bit-breiten Datentypen (LONG, FLOAT) benutzt er 32 Bit. Da der Amiga bei Systemaufrufen aber davon ausgeht, daß für jeden Parameter 32 Bit auf dem Stack liegen, benutzt der Aztec zwar standardmäßig die schnellere und speichergünstigere Lösung. Man mußte bei Systemaufrufen aber dauernd aufpassen und INTs z.B. vor der Übergabe erst in einen LONG umwandeln und bei Konstanten, wie '5', ein 'L' anhängen, um den Compiler zu zwingen, diese als LONG auf den Stack zu legen.Jetzt hat auch Lattice die Möglichkeiten zu den kurzen und schnellen 16 BitINTs eröffnet - gibt man nämlich die Option -w beim Kompilieren an, benutzt der 4.0-Compiler 16 Bit breite INTs und legt auch nur soviel auf den Stack, wie nötig ist.Daß diese zwei Details nur gemeinsam einschaltbar sind, ist zunächst verwunderlich. Das liegt aber an den Prinzipien des Compilerbaus, d.h. daran, wie ein C-Compiler zu schreiben ist. Eine Regel besagt hierzu vereinfacht, daß jede Variable und Konstante, bevor sie weiter-benutzt werden kann, zunächst einer sogenannten 'Binary Conversion', also einer Typumwandlung, unterzogen wird. Hierbei werden sämtliche ganzzahligen Datentypen (char, short, int, long) auf mindestens die Länge eines INTs erweitert. Deshalb wird jede Konstante, wenn man mit 32 Bit INTs arbeitet, bevor sie als Parameter übergeben werden kann, automatisch auf 32 Bits erweitert.Um nun die Probleme mit Systemaufrufen, wie zuvor für den Aztec beschrieben, zu umgehen, hat Lattice eine ziemlich bequeme Lösung gefunden: Man kann, wenn man definiert, daß eine Funktion extern existiert, die erwarteten Typen der Funktion zusätzlich angeben. Z.B.: 'extern long Open(char *,long)' würde definieren, daß Open einen Zeiger auf CHAR und ein LONG als Parameter benötigt. Stimmen nun die Parameter, die man im Programm übergibt, nicht mit den angegebenen Typen überein, erhält man während der Kompilation ein Warning. Vergißt man also z.B. eine 16 Bit-INT auf LONG zu erweitern, erkennt man diesen Fehler sofort an dem ausgegebenen Warning.Noch einfacher macht es der Lattice-Compiler bei Konstanten: Konstanten, die normalerweise als INT gelten, werden automatisch auf die entsprechende Länge erweitert, wenn der Parameter einer Funktion als länger definiert ist. Definiert man beispielsweise 'void Delay(long)', kann man danach, falls man 16 Bit-INTs benutzt, einfach 'Delay(10)' statt 'Delay(10L)' schreiben.

Systemfunktionen direkt

Bisher wurden die Rom Kernel-Funktionen wie ganz "normale" C-Funktionen aufgerufen, d.h. die Parameter wurden auf den Stack gelegt und dann die entsprechende Funktion aufgerufen. Assemblerprogrammierer werden wissen, daß die Funktionen intern die Parameter jedoch in bestimmten Registern erwarten. Deshalb muß man immer mit der Amiga.lib linken, die für jeden Rom Kernel Aufruf eine kleine Umsetzroutine enthält, die die Parameter vom Stack in die entsprechenden Register lädt und dann die Routine über eine Sprungtabelle, die zu jeder Library gehört, aufruft.Der Lattice-Compiler 4.0 bietet die Möglichkeit, diese Funktionen direkt mit den Parametern in den entsprechenden Registern aufzurufen, dadurch wird der Kode natürlich etwas kürzer und schneller, da die Parameter gleich in die richtigen Register gelangen und nicht erst auf dem Stack zwischengespeichert werden. Zusätzlich spart diese Methode Linkzeit, da jetzt fast keine Funktionen aus der Amiga.lib eingebunden werden müssen.Glücklicherweise muß man diese Definitionen für die Rom Kernel-Funktionen nicht selbst eingeben. Man kann einfach die entsprechenden Headerfiles aus dem Directory PROTO durch #include einbinden, die diese Definitionen enthalten und zusätzlich noch festlegen, welchen Typ die Parameter haben müssen.Sollte einmal die Kickstartversion geändert werden und es so neue Libraries geben, kann man mit einem Tool, das sich auch auf Disk 1 befindet, die sogenannten FD-Files, die z.B. mit dem Basic ausgeliefert werden, in solche PROTO-Files automatisch umwandeln. So sind auch die von Lattice mitgelieferten Files für die aktuelle Kickstartversion entstanden.

Eingebaute Funktionen

Der Compiler bietet die Möglichkeit, einige sehr häufige String- und Speicherfunktionen nicht normal aufzu-

rufen, sondern diese direkt in Maschinensprache zu implementieren. Hierbei würde dann beispielsweise für 'strlen' kein Kode erzeugt, da es sich um einen konstanten String handelt. Es würde in diesem Fall der entsprechende Wert nach D0 geladen. Auch für nicht konstante Strings verbessert sich der Kode erheblich, da die Registerinhalte ausgenutzt werden können und die C-Eigentümlichkeit der Parameterübergabe auf dem Stack wegfällt. Diese sogenannten 'built-in-functions' sind durch ein vorgestelltes '_builtin_' zu erkennen. Bindet man die Headerfiles 'string.h' und 'stdlib.h' ein, wird das '_builtin_' bei den entsprechenden Funktionen automatisch vorangestellt.

Kurzer Kode- und Datenzugriff

Die in der 3.1-Version eingeführten Optionen, um den Datenzugriff zu beschleunigen, indem man sämtliche Datenhunks (DATA und BSS) zu einem großen Hunk zusammenfügt und dann innerhalb dieses Hunks über ein konstantes Adreßregister auf die Daten zugreift, die Sprünge zwischen verschiedenen Funktionen zu optimieren und zusätzlich den Kode dadurch zu verkürzen, indem man relativ zum PC springt, sind in der 4.0-Version die Defaulteinstellung.Da hierbei aber die Sprungweite auf +/- 32k, die Größe des Datenspeichers auf 64k begrenzt ist, kann man beide Optionen getrennt abschalten.

Registervariablen

Der Lattice 4.0 unterstützt wie in der 3.1 Version 6 (!) normale Registervariablen. Es kann sich dabei um Daten beliebiger Länge bis zu 32 Bit handeln. Für diese Registervariablen werden die Datenregister D2-D7 benutzt. Leider kann er aber nur noch zwei Pointer als Registervariablen benutzen, da nur noch die Adreßregister A2 und A3 zur Verfügung stehen. Die 3.1-Version konnte immerhin 3 Adreßregister für Registervariablen benutzen.Das Problem bei der 4.0-Version liegt darin, daß die Basisadresse des Datenbereichs, die zum Adressieren nach dem schnelleren und kürzeren Datenmodell benutzt wird, jetzt in A4 gehalten wird. Früher wurde hierzu A6 gebraucht.

Linker

Der Lattice 4.0 benutzt den ehemaligen Public Domain Linker BLINK, den Lattice mittlerweile aber etwas erweitert hat. Dieser ersetzt seit der letzten Version den Standard-Amigalinker ALINK von Metacomco.BLINK ist schneller als ALINK und beherrscht ferner einige Funktionen, die mit ALINK nicht möglich sind: BLINK ve-reinigt alle Hunks mit dem Namen __MERGED zu einem einzigen Hunk, da sämtliche mit dem basisregisterrelativen Datenzugriff kompilierten Hunks diesen Namen tragen. Da ALINK das nicht tun würde, lassen sich mit dieser Option übersetzte Programme auch nicht mehr korrekt mit A-LINK linken. Des weiteren unterstützt BLINK Overlays. Das ist das Prinzip, daß ein Programm immer nur einen Teil seiner Routinen im Speicher hält und die anderen, wenn diese angesprungen werden, über zur Zeit nicht benötigte Routinen im Speicher lädt. Dadurch kann ein Großteil des Speicherbedarfs gespart werden, was vor allem bei extrem großen Programmen, die dennoch mit relativ wenig Ram auskommen sollen, wichtig ist.Ferner unterstützt die neueste Version von BLINK im Gegensatz zu ALINK Libraries mit Index. Das bedeutet, daß am Anfang jeder Library ein Inhaltsver-zeichnis ist, dem der Linker entnehmen kann, wo er welche Funktionen in der Library finden kann. Dadurch kann die Linkzeit, besonders bei sehr langen Libraries, erheblich verkürzt werden.
Die aktuelle Version des mitgelieferten Linkers kann solche Libraries zwar noch nicht erzeugen oder lesen; eine Version, die das beherrscht, ist aber für eine der nächsten Updates angekündigt.

Macroassembler

Der Lattice-Assembler ASM wurde seit der letzten Version entscheidend verbessert.Konnte man früher auf Daten innerhalb eines Files nur auf eine Weise zugreifen - d.h absolut, pcrelativ etc., was doch etwas merkwürdig war -, handelt es sich jetzt um einen vollständigen Assembler, der die normale Motorola-Syntax beherrscht. Ferner wurde die Syntax für Macros und die meisten Assemblierungsbefehle dem Amiga Assembler von Metacomco angepaßt. Der ASM versteht jetzt die von Commodore herausgegebenen Standard-Headerfiles, was das Schreiben von Programmen, die Rom Kernel-Funktionen benutzen, erheblich vereinfacht. Einige praktische Kürzel wie 'CODE' statt 'SECTION CODE' sind offensichtlich noch nicht implementiert.Es scheint auch, als sei der Assembler gerade erst fertig geworden und enthalte noch einige Bugs: Es gibt nämlich einige nicht dokumentierte Optionen beim Kommandoaufruf und bei etlichen kleinen Testprogrammen gab es bei einfachen Konstruktionen 'Internal Errors', was etwas bedauerlich ist.Da sich aber viele Programme, die eigentlich für den Metacomco-Assembler geschrieben wurden, mühelos assemblieren ließen und man davon ausgehen kann, das Lattice die noch vorhandenen Bugs so schnell wie möglich beseitigt, wird die zusätzliche Anschaffung eines Macroassemblers wie dem Amiga Assembler für Lattice-User genau wie für Aztec-Besitzer, die ja schon von Anfang an einen leistungsstarken Assembler mitgeliefert bekamen, bald überflüssig werden, da sich auch reine Maschinenprogramme mit dem ASM genausogut entwickeln lassen.

Make Utility

Das Lattice Make (LMK) ist ein Abkömmling der UNIX Utility MAKE und liegt als weiteres Tool dem Programmpaket bei.
Ein MAKE-Utility dient vor allem dazu, die Entwicklung von größeren Programmen, die aus mehreren Sourcefiles bestehen, zu erleichtern. Es kümmert sich darum, welche Files neu zu übersetzen sind. Dies geschieht mit Hilfe eines Makefiles, in dem genau festgelegt ist, welche Files aus welchen generiert werden und wie diese Generierung aussieht. Dabei wird zu jedem der zu erstellenden Files eine Liste der Programme, von denen es abhängt, angegeben. Darunter gibt man eine Reihe von Befehlen an, die ausgeführt werden, wenn das File neu erstellt werden muß. Das stellt man anhand des Datums fest: Ist nämlich eines der Files, von denen das übergeordnete abhängt, neuer als das eigentliche File, dann ist dieses nicht mehr 'up-to-date' und wird mit Hilfe der angegebenen Kom-

mandosequenz neu produziert. Die normale Anwendung ist, damit mehrere Sourcefiles mit verschiedenen Compileroptionen zu übersetzen und diese dann zusammenzulinken. Man kann aber auch gleichzeitig dafür sorgen, daß alle neuen Files abgespeichert werden, da man jeden normalen CLI-Befehl im Makefile eintragen kann.Mit Hilfe der MAKE-Utility spart man sich so eine Menge Mühe, weil man sich nicht für jedes File einzeln merken muß, wann man es zuletzt geändert hat, ab und mit welchen Optionen man es eventuell wieder kompilieren muß.

Screen-Editor

Als weitere Beigabe erhält man noch den Lattice-Screen-Editor (LSE), der recht komfortabel ist. Er unterstützt selbstverständlich die üblichen Funktionen wie Suchen, Textersatz, Blockkommandos. Man kann zwei Texte gleichzeitig bearbeiten und auch zwischen diesen hin- und herkopieren. Ferner kann man während der Benutzung Hilfstexte einlesen, die bestimmte Funktionen erklären.LSE stellt neben dem normalen Suchen noch ein Mustersuchen nach Ausdrücken, wie sie GREP (s.o) versteht, zur Verfügung. Des weiteren kann man die Fehlerausgabe des Lattice C-Compilers einlesen, um die Zeilen, die Fehler enthalten, automatisch anzuspringen.
Auch eine Macrofunktion ist implementiert, die es erlaubt, durch einen einzelnen Tastendruck bestimmte, frei definierbare Worte oder andere Zeichenfolgen auszugeben. LSE enthält noch eine Reihe weiterer Kommandos, die aufzuzählen den Rahmen dieses Tests jedoch sprengen würden.

Debugger

Als letzes erhält man mit dem Professional-Package auch den Debugger MetaScope von Metadigm, Inc., der bezüglich Leistungsfähigkeit und Komfort wohl eindeutig zu den besten gehört, die es auf dem Amiga zur Zeit gibt. Der MetaScope regelt alle Ein- und Ausgaben über Windows, von denen man jederzeit beliebig viele offenhalten kann. Es gibt hierbei verschiedene Arten von Windows. Die beiden wichtigsten Typen sind STATUS-WINDOWS, die den aktuellen Prozessorinhalt, also die Register, Flags und die gerade auszuführende Zeile in disassemblierter Form beinhalten, und MEMORY-WINDOWS, in denen man den Speicherinhalt entweder als Daten oder als disassemblierten Kode ausgeben kann.In diesem disassemblierten Kode tauchen selbstverständlich alle Symbole auf, die in dem zu debuggenden Programm enthalten waren.
Man kann die Startadresse des Speichers, der in einem Window angezeigt wird, beliebig wählen. Es können bei allen solchen Eingaben sehr komplizierte Ausdrücke eingegeben werden, die fast alle C-Operatoren enthalten dürften. Sogar die Reihenfolge der Bewertung wurde von C übernommen, was jeden C-Programmierer sicherlich freut.Es muß sich hierbei jedoch nicht um statische, also gleichbleibende Ausdrücke handeln. Man kann beispielsweise die Startadresse eines Windows, das disassemblierten Kode anzeigt, auf 'PC' setzen, d.h. es beginnt jeweils mit der Zeile, die der Prozessor gerade ausführt. Wenn man jetzt das Programm Schritt für Schritt ausführt, scrollt das Window entsprechend dem PC immer mit. Auch jedes andere Window wird andauernd aktualisiert, wenn sich z.B. etwas im Speicher geändert hat.Will man bei einem Befehl, der normalerweise eine Eingabe verlangt, einen Wert eintragen, der irgenwo in einem Window zu sehen ist, z.B. in einem Prozessorregister, was sehr häufig vorkommt, kann man diesen Wert einfach anklicken. Daraufhin wird bei dem Befehl keine Eingabe verlangt, sondern der markierte Wert wird statt dieser benutzt.Zur Kontrolle des Programmablaufes stehen mehrere Befehle zur Verfügung. Zum einen kann man das Programm einzelschrittweise laufen lassen, wobei man sich noch entscheiden muß, ob man die Unterroutinen auch im Einzelschrittmodus durchlaufen will, oder ob man das Programm durch diese normal durchlaufen läßt. ZUm andern kann man das Programm ungestört laufen lassen, muß aber Breakpoints setzen, bei denen es dann wieder anhält, damit man gewisse Speicherinhalte überprüfen kann. Hierbei kann man zusätzlich festlegen, daß eine Stelle mehrfach durchlaufen werden muß, bevor das Programm anhält, oder daß es nur dann anhält, wenn ein beliebiger, frei wählbarer Ausdruck wahr ist. So kann man das Programm anhalten, wenn z.B. ein Register einen bestimmten Wert hat.Besteht die Gefahr, daß etwas schiefgeht, kann man ein Window auch "einfrieren", was dazu führt, daß die Werte, die im Window angezeigt werden, nicht mehr geändert werden. Man kann diese Werte nach einer beliebigen Zeit aktualisieren, wieder "auftauen" oder in den Speicher oder den Prozessor zurückschreiben, wodurch sich durch einen etwaigen Fehler in diesem Bereich dann nichts geändert hat.Die Unmenge von Funktionen, die der MetaScope bietet, von denen die meisten hier garnicht erwähnt werden konnten, führt zwar dazu, daß man einige Zeit benötigt, bis man alle Möglichkeiten dieses Debuggers wirklich ausschöpfen kann. Wenn man sich aber eingewöhnt hat, steht einem mit dem MetaScope ein Tool zur Verfügung, das eine sehr schnelle und relativ leichte Fehlersuche ermöglicht.

Fazit

Der Lattice 4.0 ist ein gelungener Nachfolger der bisherigen Lattice-Versionen, die an einigen Punkten verbessert wurden und auch neue Konzepte, wie das direkte Anspringen von Rom Kernel-Funktionen und built-in-Funktionen, einbringt. Die Mitlieferung eines vollwertigen Macroassemblers ist ein begrüßenswerter Schritt, da man bei einigen Anwendungen nicht darauf verzichten kann, bestimmte Routinen der Geschwindigkeit wegen in Assembler zu programmieren. Man kann sicher davon ausgehen, daß die noch vorhandenen Bugs bald beseitigt sein werden.Das gesamte 'Professional Package' enthält viele nützliche Tools, von denen vor allem der MetaScope Debugger eigentlich unentbehrlich ist. Ob sich die doch um einiges höhere Ausgabe für die restlichen Tools für jedermann lohnt, ist trotzdem fraglich. Vor allem die Text-Utilities, die Make-Utility und der Editor sind eher für semi-profesionelle oder professionelle Anwendung gedacht, da diese beim Schreiben von kleineren Programmen nicht sonderlich wichtig, wenn auch trotzdem nützlich, sind.Bleibt nur abzuwarte, wie die neue Aztec-Version ausfällt.

Hersteller: Lattice Californien.
Anbieter: Philgerma-München.
Preis: 498.- Grundversion
798.-incl.Screeneditor, Make,Degugger

VON JOBST HERMEIER

DER BUCH HALTER

AMIGACALC im Test

Rechtzeitig zur Bescherung schneite noch AMIGACALC auf den redaktionellen Gabentisch. Wir testeten, ob man damit auch über den Frühling kommen kann, und ob das offizielle COMMODORE-Logo auf dem Titel auch hält, was es verspricht.

Aha !

Mit einem schmatzenden "Knirsch" springt die Verpackung auf. Heraus fallen eine Diskette und ein 130seitiges Booklet, die Dokumentation in deutsch. Das COMMODORE-Emblem auf dem Einband bedeutet aber nicht, daß das Programm dort auch entwickelt wurde. Dies besorgte die Schweizer Softwarefirma DOS GmbH aus Basel. Und was kann man damit nun anfangen ? "CALC"-ulieren, wie der Name schon sagt. AMIGACALC gehört nämlich zu der Kategorie der reinen Tabellenkalkulationsprogramme.

Aller Anfang ..

Nach dem Hochfahren erinnert das sich bietende Bild ein bißchen an das gute alte Multiplan auf dem PC. Zeilen und Spalten sind numerisch fortlaufend bezeichnet. Bei diesem Verfahren erfordert die Angabe einer Zellposition immer mindestens zwei Zahlen und zwei Buchstaben (zum Beispiel Z3S4 für Zeile 3 Spalte 4). Im Gegensatz dazu werden bei den nahen Verwandten, den Spreadsheets, die Spalten durch Buchstaben gekennzeichnet, und es braucht deshalb minimal nur zwei Zeichen zur Zellidentifikation (zum Beispiel A1).

Die eigentlichen Probleme fangen aber erst bei dem Versuch an, etwas einzugeben. Es gibt nämlich keine Stelle auf dem Bildschirm, an der die Eingaben erscheinen, und die ersten zaghaften Tastenanschläge laufen ins Leere. Das zwingt zu einem Blick ins Handbuch, der ja eigentlich schon vor diesen Bemühungen hätte stattfinden sollen. Dort liest man nach einigem Blättern, daß es sogar zwei Möglichkeiten gibt, Datenwerte in eine "Blattzelle" zu befördern. Zum einen löst ein Klick auf der gewünschten Zelle den Requester "Eingabe" aus, in dem man den Datentyp des Werts (Zahlen,Texte,Formeln) wählen und ihn endlich eingeben kann. Zum anderen kann man in einem Pull-Down-Menü das Eingabefenster aktivieren, das dann bis zum Schließen auf dem Bildschirm verbleibt und eine ganze Zeile zur Eingabe und Edierung bereitstellt. Die Handhabung des Programms wird durch dieses "Verstecken " der Eingabemöglichkeiten allerdings etwas umständlicher, und dem Anfänger wird der Einstieg erschwert.

Ausrüstung

Hat man diese Schwierigkeiten einmal überwunden, kann man an die eigentliche Arbeit, nämlich das Erfassen von Zahlen und Formeln, gehen. Die einzelnen Zellen des Blattes lassen sich dazu auf vielfältigste Weisen gestalten. Außer den üblichen Formaten wie : Textjustierung, Zahl der Nachkommastellen, Exponential- oder Datumsdarstellungen bietet AMIGACALC noch einige Besonderheiten. Beispielsweise kann eine Zelle als Debitor/Kreditor formatiert werden. Beides sind Begriffe des Rechnungswesens und bedeuten bei AMIGACALC, daß je nach Vorzeichen der Zahl eine bestimmte Buchstabenkombination vorangestellt wird. Voreingestellt sind DB als Debitor und CR für Kreditorenwerte.

Globalere Einstellungen kann man mit dem Kalkulationsformat vornehmen. Hier findet sich zum Beispiel eine Tabelle, mit der sich Bezeichner und Umrechnungsfaktoren für Einheiten (etwa $ versus DM) festlegen lassen. Wie bei ähnlichen Amigaprogrammen dieser Art (z.B. Haicalc) läßt sich auch bei AMIGACALC die Richtungswirkung der RETURN-Taste bestimmen. Ist das Eingabefenster geöffnet, bestimmt diese Einstellung, wohin der Zellcursor nach jeder Betätigung der RETURN-Taste springt (zum Beispiel: nach unten, nach rechts usw.).

Funktional

Schaltet man bei der Eingabe von Zahl/Text auf Formel um, enthält die Menüleiste die Aufzählung aller Funktionen; und das sind schon einige. Außer den Standardfunktionen für trigonometrische, statistische oder numerische Berechnungen gibt es zusätzliche für Datums- , logische oder finanzmathematische Ausdrücke. Beim Durchforsten der Funktionsnamen sind mir allerdings einige etwas merkwürdig vorgekommen. So heißt die Funktion zum Feststellen des Minimums eines Zellbereichs MINI, während man die Minuten eines Uhrzeitwertes mit MIN bestimmen kann. Selbst in mathematischen Lehrbüchern wird aber das Minimum - wenn kein Symbol dafür verwendet wird - durch die Abkürzung MIN ausgedrückt. Die meisten Leute, die eine AMIGACALCformel mit MIN zu Gesicht bekommen, werden wohl aufs Glatteis geführt. Das gleiche gilt für die Funktion GANZ, die den

kleinsten Integerwert einer reellen Zahl ermittelt, aber überall sonst als INT bekannt ist. Diese zwei Beispiele zeigen stellvertretend, wie es sich mit Programmtexten und Dokumentationen in deutscher Sprache verhält. Ist ein AMIGA-Programm nur in englisch erhältlich, bekommt es gleich Minuspunkte in der Wertung verpaßt. Hat man dann endlich wieder mal eins in deutsch, werden die Texte gleich um so kritscher betrachtet, da man über die kleinen Fehler in der Muttersprache viel eher stolpert als über die Faux pas in einer Fremdsprache.

Besonderheiten

Denjenigen, die immer nach etwas Besonderem Ausschau halten, bietet AMIGACALC auch ein paar Schmankerln an. So etwa eine Zeichensatzerweiterung, die vom IBM-PC bekannte Linien und Doppellinien mit dazugehörigen Ecken und Kreuzungen bereitstellt. Dadurch werden Zahlenkolonnen erst zu richtigen Tabellen und - wie man Abbildung 1 entnehmen kann - sind sie auch für andere Zwecke vielseitig verwendbar. Mit einem 24-Nadeldrucker lassen sich diese Verzierungen anstandslos zu Papier bringen. Zum Thema Drucker gibt's ebenfalls etwas anzumerken. AMIGACALC weist zusätzlich zu den Betriebssystemeinrichtungen zum Drucken nämlich einen eigenen Druckertreiber auf. Damit kann (oder muß) man jede erwünschte Steuersequenz des jeweiligen Geräts mit Hilfe eines Editors einstellen. Auf diesem Wege kommt auch der Ausdruck der Linien zustande, wobei einfach der im Drucker vorhandene IBM-Zeichensatz ausgenutzt wird. Ein Vorteil dieser Technik ist, daß man sich ein eigenes Emblem als Druckerzeichen definieren kann.

Als weiteres Extra erhebt AMIGACALC die Kalkulationen auf Wunsch in die dritte Dimension. Dazu lassen sich mehrere BLÄTTER - also Tabellen - zu einem MODELL kombinieren. Dieses Feature läßt sich zum Beispiel bei der Planung gut einsetzen. Hier gibt es in der Regel ein Kalkulationsblatt gleichen Aufbaus für Letzjahreswerte, Daten dieses Jahres und eben Plandaten. Zu Vergleichen lassen sich dann Werte aus allen Blättern in einer Rechenformel verwenden.

Erwähnung verdient haben zuletzt noch die Macrofähigkeiten. Damit lassen sich Befehlssequenzen, die normalerweise per Hand eingegeben werden müßten, zu automatisierten Abläufen zusammenfassen.

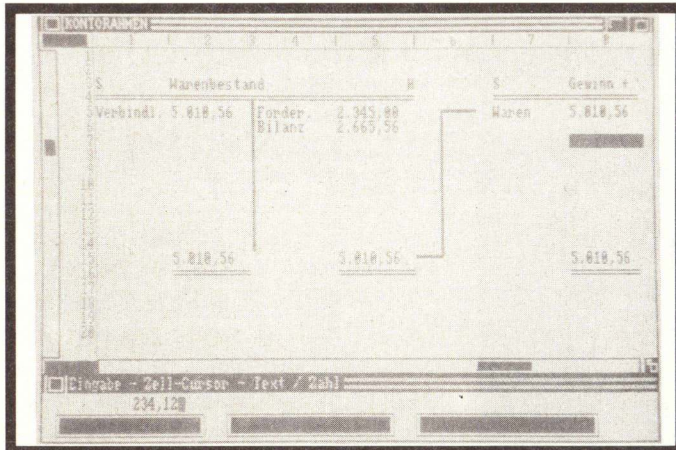


Bild 1: Buchführung wie sie sein muß

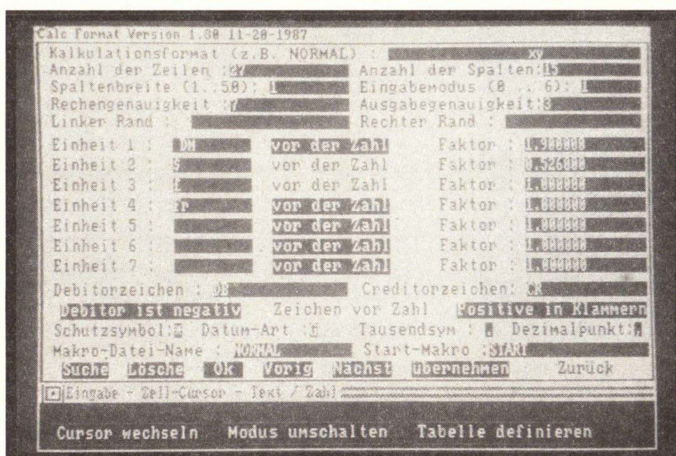


Bild 2: AMIGACALC hat vielfältige Einstellungsmög l-ichkeiten

Unterm Strich

AMIGACALC ist sicher eher den braven Arbeitstieren als den Vollblütern zuzurechnen. Nichtsdestotrotz besitzt es alle Optionen, die zur Abwicklung ernsthafter Aufgabenstellungen benötigt werden. Die auf dem Einband als "leicht und problemlos" beschriebene Bedienung ist meiner Meinung nach aber recht gewöhnungsbedürftig. Mit der auf dem Amiga ja zum Begriff gewordenen Intuition kommt man hier nicht sehr weit. Dafür ist aber das Handbuch leserlich und übersichtlich gestaltet, was manchen Fehltritt wettmachen kann. In der Ausführlichkeit läßt es teilweise wieder zu wünschen übrig, so fehlt zum Beispiel die Beschreibung einiger Funktionen (QSUMME, QMITTEL) völlig.

Vermissen wird der verwöhnte Amigafreak auch die üppigen Grafikmöglichkeiten anderer Programme; bei AMIGACALC wird man dafür mit einer Pseudografik aus Sternen abgespeist. Das Programm ist aber allen zu empfehlen, die nicht mitten in ihrer Steuererklärung einem GURU begegnen möchten, da die vorhandenen Funktionen im Test immer sicher ausgeführt wurden. Mit einem Preis von ca. 248.- DM bleibt AMIGACALC noch im erschwinglichen Bereich, obwohl einige vergleichbare Produkte in dieser Hinsicht eher Zeichen setzen.

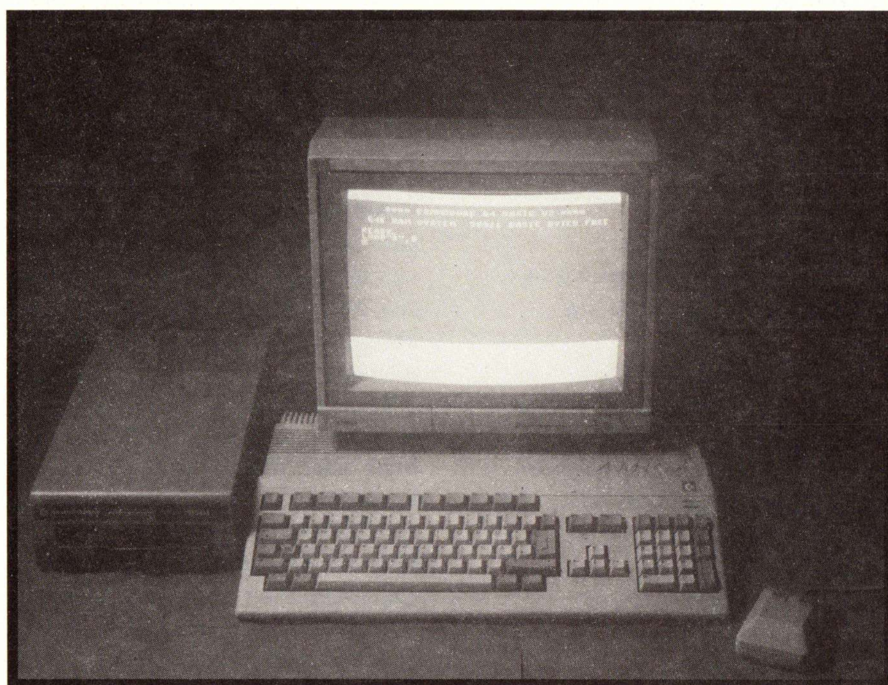
Plus/Minus-Kasten

+ Dreidimensionale Kalkulationen
+ Erweiterter Zeichensatz
+ Macrofähigkeiten
+ Zuverlässig
- Mindestens 1 MB Speicher
- Keine Graphik
- Schwierige Druckeranpassung

VON ANDREAS KRÄMER

DIE GUTE ALTE ZEIT

The 64 Emulator



Vor nicht allzulanger Zeit wurde in der KICKSTART bereits ein Emulator, der den guten alten Commodore Heimcomputer emuliert, vorgestellt, das Ergebnis war nicht gerade berauschend. Frisch aus Amerika erreichte uns ein weiterer Emulator, der auf den schlichten Namen 'The 64-Emulator' getauft ist. Lohnt es sich, den Emulator zu kaufen, oder sollte man sich das mittlerweile günstig zu erhaltende 'Orginal' anschaffen ? Wie steht es mit der Kompatibilität ? Mit welcher Geschwindigkeit arbeitet der Emulator ? Diese und andere Fragen beantwortet der Test.

Was gibt's fürs Geld ?

Im Lieferumfang befinden sich neben der Emulator-Diskette ein dünnes, 16 Seiten starkes Handbuch und ein serielles Kabel, das in den parallelen Port des Amiga gesteckt wird und den Anschluß der 1541-Floppystation ermöglicht - natürlich nur, wenn vorhanden.

Ans Werk !

Besitzer eines AMIGA 1000 müssen selbstverständlich zuerst die Kickstart laden, anstatt der Workbench-Diskette wird dann die 64er-Emulator Diskette eingeschoben. Nach kurzer Wartezeit und einem Intro-Bildschirm des Herstellers (ReadySoft Inc.) erscheint das vielen vertraute, (fast) originale Einschaltbild des erfolgreichsten Heimcomputers aller Zeiten.
Das erste kleine BASIC-Programm wird eingetippt und gestartet.Läuft, gut! Also ans Werk, die alte Programm-Sammlung und die 64er Floppy aus dem Schrank geholt angeschlossen.

Wie kompatibel ?

Als erstes fiel mir eine Diskette mit einer reinen BASIC-Programmsammlung in die Hände. In die 1541-Floppy geschoben, den üblichen Verzeichnis-Lade-Befehl eingegeben(LOAD"$",8) und Return gedrückt: die Floppy 'springt an', und nach kurzer Wartezeit signalisiert der Emulator mit einem Ready, daß der Ladevorgang beendet ist. Mit LIST wird das Disketten-Verzeichnis ausgegeben. Ein 36-Block-langes Programm wurde ausgewählt

und gestartet. Nach scheinbar unendlich langer Wartezeit starte ich das Programm mit RUN. Das mit einigen POKEs gespickte Programm beginnt seinen Ablauf ohne Probleme, nur langsamer. Verschiedene Tests von

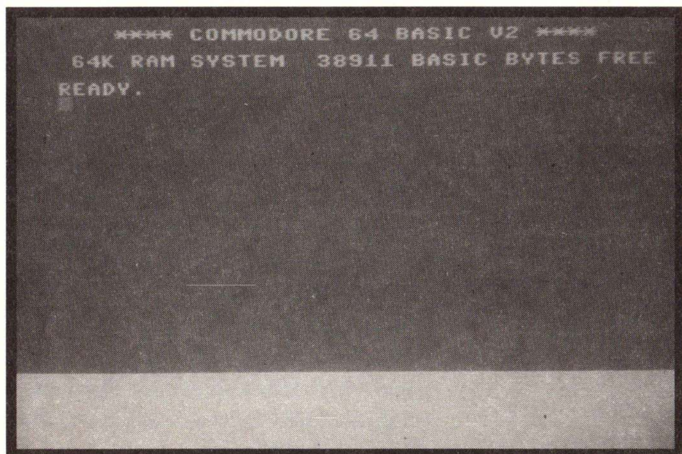


Bild 1: Nach einem Reset meldet sich der Emulator wie folgt...

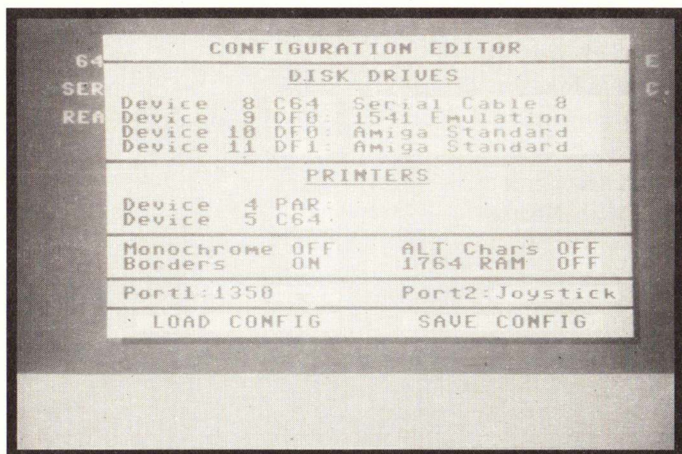


Bild 2: Der Konfigurations-Editor ermöglicht eine Vielzahl von Einstellungsmöglichkeiten.

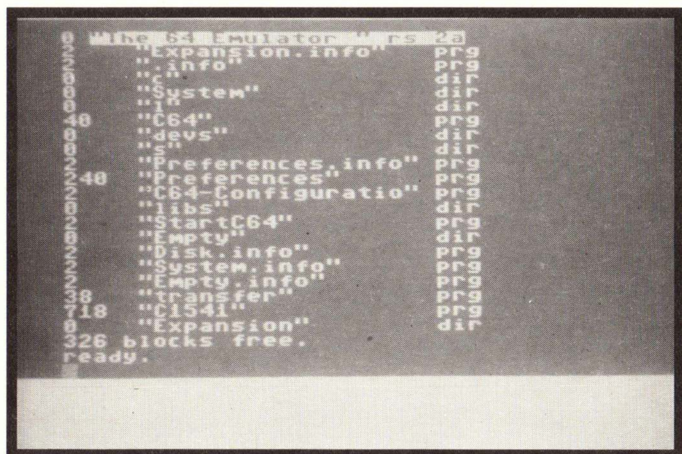


Bild 3: Die Emulator-Diskette enthält die im Bild aufgezeigten Dateien, wobei das Verzeichnis vom Emulator gelesen wurde.

BASIC-Programmen verliefen positiv. Aber in BASIC geschriebene Programme sind nicht der Weisheit letzter Schluß, auch Maschinenprogramme müssen zeigen, ob sie laufen. In der Sammlung gekramt und die Spiele-Disketten zum Vorschein geholt. Doch die große Enttäuschung ließ nicht auf sich warten. Als Resümee kann man sagen, daß der Emulator mit Spielen nicht viel am Hut hat. Entweder wollten die Spielprogramme garnicht laufen (die meisten), oder, wenn sie liefen, war die Geschwindigkeit schier berauschend. Die ruckelnden Sprites ließen keine rechte Freude aufkommen. Doch wie sieht die Sache mit Anwender-Software aus ? Zunächst wurde das bekannteste Textverarbeitungsprogramm ausgewählt - Vizawrite. Hierzu muß gesagt werden, daß keine Funktionseinschränkung erkannt werden konnte. Global kann im Anwender-Bereich eine hohe Kompatibilität anerkannt werden, genauso verhält es sich mit reinen BASIC-Programmen. Zum Spielen dient der Emulator nicht, zu viele Produkte versagen den Dienst.

Ein Konfiguration-Editor

Als Besonderheit ist dem Emulator ein sogenannter 'Configuration Editor' mitgegeben, der immer über eine bestimmte Tastenkombination angesprungen werden kann. Hier sind unter anderem verschiedene Laufwerkskonfigurationen einstellbar. So ist es möglich, die 3.5" Laufwerke des AMIGA als 64er-Speichermedium zu verwenden. Doch damit nicht genug, sämtliche Laufwerke können vom 64er angesprochen werden, angefangen von df0 bis hin zur df3-Geräteadresse. Dabei stehen dem Anwender zwei Optionen zur Auswahl: eine 1541-Emulation, wo 170 KByte formatiert werden und die Daten im 64er-Modus abgelegt und gelesen werden; und eine Standard-AMIGA-Emulation, wo 880 KByte zum Beschreiben bereit stehen. Die Daten werden im AMIGA-DOS-Format abgelegt und dementsprechend gelesen. Der Anwender sollte reine AMIGA- und 64er- Disketten trennen, da beim Ausgeben des Verzeichnisses alle auf der Diskette befindlichen Programme angezeigt werden, und somit leicht Verwirrung auftreten kann. Wer jetzt glaubt, er müßte die teuren 3.5"-Disketten für den 64er 'verschwenden', liegt zwar richtig, sollte aber bedenken, daß die kompletten 880-KByte des AMIGA-DOS-Formates für 64er-Programme zur Verfügung stehen. Natürlich lassen sich auch 3.5"-Disketten im reinen 64er-Format erstellt werden. Von AMIGA-DOS ist es möglich sich 64er Programme mit dem CLI-Kommando 'type <Datei> opt h' anzuschauen. Nicht nur die erhöhte Speicherkapazität der Diskette im AMIGA-DOS-Format ist ein interessantes Feature, auch durch Ablegen der Programme im AMIGA-DOS-Format ergibt sich eine Ladegeschwindigkeitssteigerung um den Faktor drei. Außerdem ist es möglich, Festplatten vom 64er aus zu betreiben, sicherlich eine interessante Angelegenheit.

Drucker

Desweiteren besitzt der Editor noch verschiedene Möglichkeiten, einen Drucker anzusteuern. Hier stehen ein paralleler und ein serieller AMIGA-Drucker sowie ein Commodore 64-serieller Drucker zur Auswahl.

Von Farbe, Zeichensätzen und Speichererweiterungen

Im Editor kann die Farbe ausgeschaltet werden, laut Beschreibung ergibt sich dadurch eine höhere Ablaufgeschwindigkeit der Programme. In der Praxis ist der Zuwachs aber als gering zu bewerten.

Bedingt durch die höhere Auflösung des AMIGA gegenüber dem 64er konnten die Programmierer ein weiteres Feature mit einbauen: Alternative Zeichensätze.

Eine 1764 RAM-Speichererweiterung wird auf Wunsch emuliert, allerdings muß in diesem Falle 1-MByte Speicher vorhanden sein.

Die letzte Einstellungsmöglichkeit besteht im Verändern der Game-Ports. Joystick, Paddle, 1350 und 1351 stehen zur Auswahl.

Selbstverständlich lassen sich im Editor gemachte Änderungen speichern bzw. wieder laden.

Die Tastatur

Alle besonderen Commodore 64-Tasten werden von der AMIGA-Tastatur emuliert. Erwähnenswert ist lediglich, daß alle Cursor-Tasten belegt sind, und somit ein einfacheres Positionieren des Cursors ermöglicht wird (wir erinnern uns).

Fazit

Die vielen Extras sind sicherlich einiges wert, aber immer noch stellt sich die grundlegende Frage, ob sich die Anschaffung des Emulators lohnt, besonders wenn man den niedrigen Preis des Commodore 64 bedenkt. Bereits für 300.- DM erhält man das Gerät in einigen Läden, und mit Kompatibilitätsproblemen braucht auf keinen Fall gekämpft zu werden. Der Emulator besitzt letztendlich doch einige erhebliche Schwächen. Die Legende vom 'Spielewunder 64' erfährt mit dem Emulator auf dem AMIGA jedenfalls keine Wiedergeburt.

Anwendungssoftware läuft zum größten Teil, soweit auf komplizierte Grafik verzichtet wird. Doch es gibt in diesem Bereich weitaus bessere Programme für den Amiga - also wofür ? Vielleicht für diejenigen, die viel Software für den 64er besitzen und sich keine teure AMIGA-Software kaufen können oder wollen.

Sicherlich auch für Leute, die Daten vom 64er zum Amiga zu transferieren gedenken oder einfach nur Spaß daran haben, einen Computer in einem Computer zu besitzen.

Der Preis ist mit 149.- DM akzeptabel. Probleme mit Speichererweiterungen konnten nicht festgestellt werden.

Hersteller:
Ready Soft Inc.
New York
Anbieter: IM
Tel. 069-7071102
Preis: 149.- DM

Plus/Minus-Kasten

+ viele Einstellungsmöglichkeiten über einen Editor
+ 3.5"-AMIGA-Disketten können sowohl im 64er als auch im AMIGA-DOS-Format verwendet werden
+ höhere Lade- und Schreibgeschwindigkeit bei der Verwendung des AMIGA-Formates
+ fast alle BASIC-Programme lauffähig

- langsame Arbeitsgeschwindigkeit
- geringe Kompatibilität bei Maschinenprogrammen, besonders bei Spielen
- dünnes Handbuch in englischer Sprache
- im Vergleich zum Gerät hoher Verkaufspreis

KICKS FÜR INSIDER

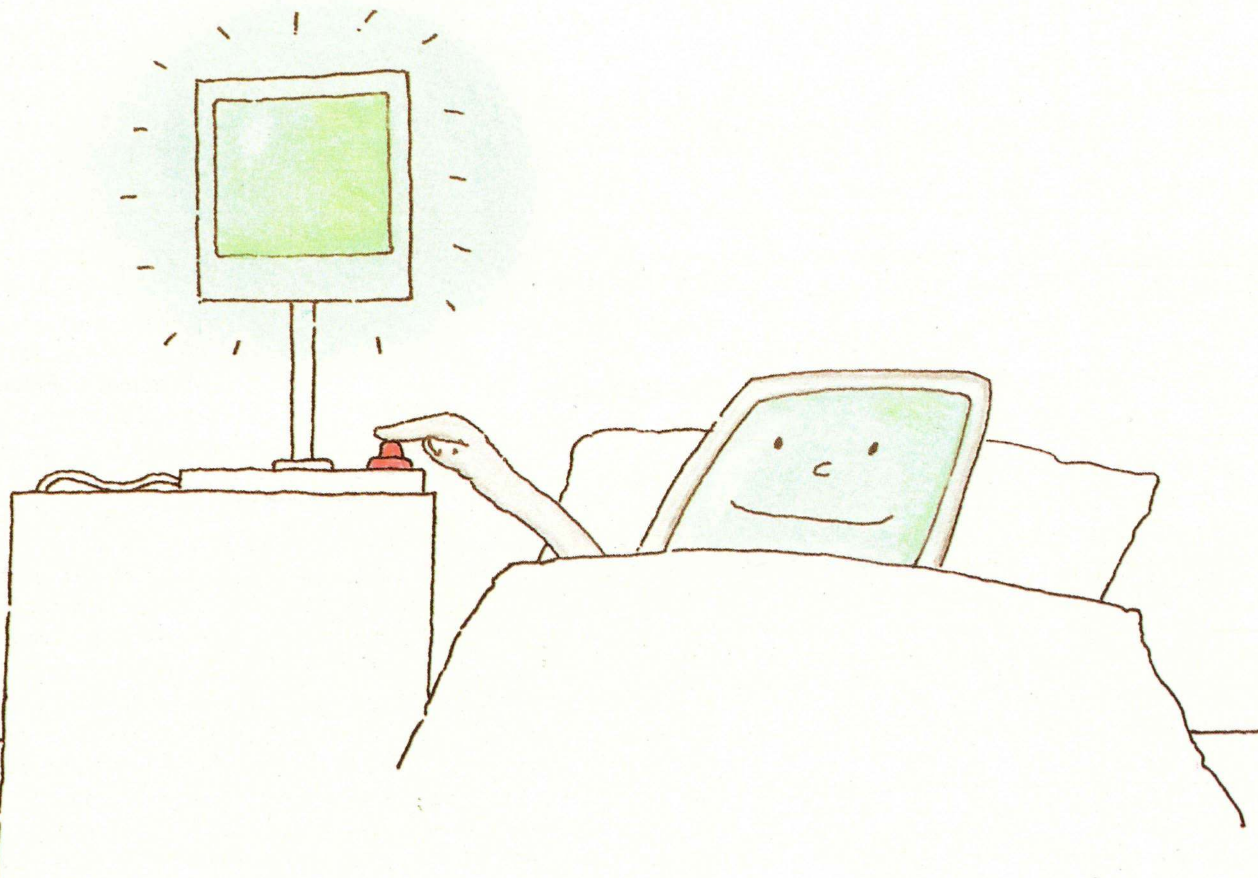


Wir freuen uns, Sie zur zweiten Ausgabe der KICKS begrüßen zu dürfen. Auch diesmal haben wir interessante Programme zusammengepackt, die zum einen nützliche Hifen im Programmiereralltag darstellen, zum anderen aber auch zeigen, wie man eben diese Probleme programmiertechnisch löst. Vergessen Sie bitte nicht, daß die KICKS größtenteils von unseren Lesern gestaltet werden. Wir bitten Sie deshalb um rege Beteiligung.

INHALT

VON GERALD CARDA

DER RÖHREN SCHONER



Da kaufen Sie einen Rechner und einen Monitor für teures Geld und müssen feststellen, daß ersterer dem letzteren langsam aber sicher die Bildschirmmaske Ihres Lieblingsprogramm einbrennt, während Sie nach einem außerplanmäßig langem Telefongespräch oder einer anderen zwischenmenschlichen Interaktion an den Ort der Bits und Bytes zurückkehren.

An diesem Punkt hilft Ihnen das hier gedruckte Prógrämmchen weiter. Es schaltet die Videoausgabe des Rechners nach einer vorgegebenen Zeitspanne ab und verlängert auf diese Weise die Lebensdauer der Ihnen eigenen Bildröhre. Natürlich überwacht das Programm während der Dunkelzeit eventuell vorkommende Tastatur- oder Mausereignisse und schaltet bei deren Eintreffen die Bildausgabe wieder an.

Grundsätzliches

Die Multitasking-Philosophie des Amigabetriebssystems verlangt bei der Programmierung solcher Routinen die Beachtung einiger Grundregeln. So muß das Programm so wenig Rechenleistung wie möglich verbrauchen, um den vorrangig laufenden Anwenderprogrammen keine Prozessorzeit zu stehlen. Auch sollte es vermieden werden, auf feste Adressen zuzugreifen, die sich mit der nächsten Auflage des Rechners verschieben könnten. Die einzige von Commodore dokumentierte Adresse ist die Speicherstelle $4. In ihr steht der Basisvektor auf den Systemkern Exec.

Kommen wir jedoch wieder auf das angekündigte Programm zu sprechen. Es ist der besseren Systemzugänglichkeit wegen in der Sprache C geschrieben. Daß auch diese Sprache trotz der vielen Lobpreisungen nicht alle Wege zum Betriebssystem öffnet, zeigt die Notwendigkeit der kurzen Assemblerroutine, die in diesem speziellen Fall erforderlich ist. Fangen wir aber der Reihe nach an.

Modularer Aufbau

Die ersten Zeilen enthalten wie gewohnt die benötigten Includedateien. Darauf folgt die Definition einiger wichtiger, im Programm benötigter Konstanten. Die Konstante TIMEINTERVAL enthält den Zeitraum, nach dem die Tastatur und die Mausereignisse abgefragt werden. Die Programmierung ist so vorgenommen, daß der Task in der Zwischenzeit ruht und so keine Zeit verbraucht. Das Programm erhält mit der hier gewählten Konstante nur alle vollen Sekunden den Prozessor. Dieser Wert ist ein gutes Mittel aus Ansprechzeit und Systembelastung durch den Task. Unter Ansprechzeit ist die Zeit zu verstehen die das Programm maximal benötigt, um nach einer Dunkeltastung die Bildausgabe wieder einzuschalten. Da das Programm nur alle TIMEINTERVAL-Zeiteinheiten reagieren kann, wird ein Ereignis, das kurz nach der Prozessorübergabe eingetreten ist, erst nach der Zeitspanne TIMEINTERVAL mit der dann folgenden Prozessorübernahme berücksichtigt.

Die zweite Konstante stellt die Anzahl von Zeiteinheiten der Länge TIMEINTERVAL dar, nach denen die Bildausgabe gestoppt wird. Da der voreingestellte Wert von TIMEINTERVAL eine Sekunde ist, ist der Wert von LIMIT genau die Anzahl von Sekunden, nach denen das 'Licht', im wahrsten Sinne des Wortes, ausgeht. Sie können für beide Konstanten eigene, ganzzahligen Vielfache einer Sekunde einsetzen. Um auch Bruchteile von Sekunden verwenden zu können, ist die Funktion WaitForTimer() zu ändern.

Der Wecker

Diese Funktion übergibt in den Strukturelementen tv_secs und tv_micro die genaue Zeit, die der Task bis zur nächsten Prozessorübernahme warten soll. Diese Weckerfunktion wird über das Timer.device mit dem Befehl TR_ADDREQUEST erreicht. Die Ausführung des Befehls wird durch DoIO() herbeigeführt.

Erst gehen die Lichter aus ...

Die dann folgende Funktion ist für das Dunkelschalten des Bildschirms zuständig. Da das Programm gemäß der oben erwähnten Grundsätze auf keinerlei Hardware-abhängige Adressen zugreifen sollte, habe ich hier einen kleinen Trick angewandt. Durch das Gfxmacro OFF_DISPLAY wird die für die Bildschirmdarstellung nötige Bitplane-DMA abgeschaltet. Somit ist der Bildschirm zwar leer, besitzt aber immer noch die eingestellte Hintergrundfarbe, wie zum Beispiel die Commodorespezialfarbe 'Workbench-Blau'. Dieses Problem wird auf eine besondere Art gelöst. Die Funktion Dunkel() öffnet einen eigenen Screen über dem momentan aktuellen mit der Funktion OpenScreen(). Der so entstandene neue 'Hintergrund' erhält durch die Funktion SetRGB4() die Hintergrundfarbe Schwarz. Der darauffolgende Aufruf von OFF_DISPLAY erzeugt dann den gewünschten schwarzen Screen auf schwarzer Röhre. Am Ende der Funktion wird dann noch ein Flag gesetzt, das innerhalb der Main() Routine ausgewertet wird.

...und dann wieder an

Der Zweck der Funktion Hell() ist genau der, auf den der Name schließen läßt. Sie 'schaltet' die Bildausgabe wieder ein. Dazu wird zuerst der 'darkscreen' geschlossen und dann die Bitplane-DMA mit ON_DISPLAY wieder eingeschaltet.

Das Hauptprogramm

Die Main() Funktion in dem hier vorgestellten Programm sieht etwas gewöhnungsbedürftig aus. Die hier gewählte Art der Programmierung mit dem Einsprung über void _main() und das Ende über XCEXIT() verkürzen die Länge des fertigen Programms in diesem Fall um fast 50%. Auf einer fast vollen Workbench kann diese Tatsache schon darüber entscheiden, ob das Programm auf derselbigen heimisch wird, um seinem User zu dienen (hat hier zufällig einer Tron gesehen ?).

Die Funktion Main() geht erst einmal die ausgetretenen Pfade des Libraryöffnens. Danach werden alle Vorbereitungen zum Öffnen des Input.device und des Timer.device getroffen und durchgeführt. Über das Timer.device erfährt das Programm von der vergangenen Zeit und über das Input.device über eventuell inzwischen eingegangene Eingabeereignisse. Dazu wird ein spezieller Weg eingeschlagen. Über den Gerätebefehl IND_ADDHANDLER wird ein Eingabeereignisverwalter, vielleicht besser Inputhandler genannt, in die Reihe der schon vom System vorhandenen eingeklinkt. Der neue Handler erhält hierbei die Priorität 51 und liegt dadurch vor den Handlern von Dos und Intuition. Auf die genaue Funktion eines solchen Handlers gehe ich gleich ein.

Als nächstes kommt die Fast-Endlos-Schleife ('Fast' wie beinahe und nicht wie Chip) mit der Abfrage nach Control-C. Die Konstante CONTROL_C ist am Anfang des Listings definiert, wird durch Drücken der Tastenkombination Ctrl und C wahr und erlaubt so den Abbruch des Programms. Da dieses jedoch meistens als Task mit dem CLI-Befehl Run gestartet werden wird, reagiert die Abfrage auch auf den

Break-Befehl des CLI. Wenn Sie das Programm mit dem Namem Blank durch

‘1> Run Blank’

starten und sich anschließend mit dem Befehl Status die Nummer des Tasks ansehen, können Sie durch Eingabe von Break, gefolgt von der angezeigten Tasknummer, das Programm wieder aus dem Speicher entfernen.

Innerhalb der Schleife wird als erstes eine Einheit TIMEINTERVAL gewartet. Dann wird das Flag Ereignis abgefragt, um festzustellen, ob eine Taste gedrückt wurde oder die Maus die Position verändert hat. Ist dies nicht der Fall, wird die verstrichene Zeit in der Variablen Sekunden um eins erhöht und mit LIMIT verglichen. Ist die maximale Zeit erreicht, wird die Funktion Dunkel() aufgerufen. Ist ein Ereignis eingetreten, wird der Teil innerhalb von else ausgeführt.

Das Ende

Tritt der eben beschriebene Break auf, schließt das Programm alle benötigten Devices und Libraries und gibt den Speicherplatz an das System zurück. Als letztes folgt der Rücksprung zum DOS mit dem Fehlerkode 20.

Assembler ist schneller

Soweit der C-Teil des Programms. Der Inputhandler ist in Assembler geschrieben, um die volle Kontrolle über die benutzten Register zu behalten und die Ausführung der Routine auf ein Mindestmaß an Zeit zu beschränken. Wenn Sie zum Beispiel die Maus in der Sekunde um 1000 Punkte bewegen (einmal schnell nach links und zurück), wird diese Routine 1000 mal aufgerufen. Folgende Übergabeparameter können beim Ansprung der Routine durch das Betriebssystem als gesichert gelten; das Register A0 enthält einen Zeiger auf das eingetretene Eingabeereignis und A1 enthält einen Zeiger auf den Datenbereich der Routine. Dieser Datenbereich wurde zuvor mit Hilfe der Interruptstruktur handlerStuff im Element is_Data auf &me[0] festgelegt. Der so zugängliche handlereigene Datenbereich wird hier jedoch nicht ausgenutzt. Die Routine adressiert über A0 die Variable ie_Class des aufgetretenen Eingabeereignisses. Innerhalb dieser Variablen ist die Art des Ereignisses abgelegt. Der Zahlenwert eins steht für ein Tastaturereignis und der Wert zwei für eine Mausbewegung. Ist eins dieser beiden Ereignisse eingetroffen, wird die Variable ‘ereignis’ auf diesen Wert gesetzt und dem Hauptprogramm so eine Mitteilung gemacht. Da, wie oben besprochen, noch mehr Inputhandler im System sind, muß im Register D0 ein Zeiger auf das nächste zu verarbeitende Eingabeereignis an den folgenden Handler übergeben werden. Da der hier vorgestellte Handler das Ereignis nicht verarbeitet sondern lediglich überprüft, gibt er den anfangs erhaltenen Zeiger an den nächsten Handler weiter.

Viel Spaß mit dem Röhrenschoner

Der Make-Text

```
Um nun eine lauffähige Version des Programms zu erhalten, können Sie a) folgende
Schritte durchführen:

1) Eingeben der beiden Listings

2) Kompilieren des C-Teils durch
   lc -v blank
   Die Option -v schaltet die Stacküberprüfung durch den Compiler aus.

3) Assemblieren des Maschinenspracheteils durch
   asm blank.asm

4) Binden der erzeugten Objektdateien durch    blink lib:c.o,blank.o,handler.o to
blank   lib lib:lc.lib,lib:amiga.lib

   (Der Aufruf von Blink muß in einer Zeile in den Rechner eingegeben werden. Der
hier entstandene Zweizeiler ist drucktechnisch bedingt.)

Oder b) Sie wollen den Quelltext nicht abtippen bzw. haben die nötigen Werkzeuge
zum Kompilieren und Assemblieren nicht, so besteht wie immer die Möglichkeit die
Listings und die fertigen Programme dieser Ausgabe auf Diskette zu bekommen. Sie
können sie einfach bei der Redaktion bestellen.
```

```
ie_Class     EQU      4

      CSECT    text

      XREF     _ereigniss
      XDEF     _HandlerInterface

_HandlerInterface:
      clr.l    d0
      move.b   ie_Class(a0),d0
      cmpi.b   #1,d0
      bne.s    no_rawkey
      move.l   d0,_ereigniss
no_rawkey:
      cmpi.b   #2,d0
      bne.s    no_rawmouse
      move.l   d0,_ereigniss
no_rawmouse:
      move.l   a0,d0
      rts
      END

```

Der Eingabe-Handler in Assembler

```
#include <exec/types.h>
#include <exec/ports.h>
#include <exec/memory.h>
#include <exec/io.h>
#include <exec/tasks.h>
#include <exec/interrupts.h>
#include <intuition/intuition.h>
#include <graphics/gfxbase.h>
#include <graphics/gfxmacros.h>
#include <devices/input.h>
#include <exec/devices.h>
#include <devices/inputevent.h>
#include <hardware/custom.h>
#include <hardware/dmabits.h>

#define TIMEINTERVAL 1
#define LIMIT 60
#define CONTROL_C ((SetSignal(0,0x1000)&0x1000))

extern struct MsgPort *CreatePort();
extern struct IOStdReq *CreateStdIO();
extern void HandlerInterface();
extern struct Custom custom;

struct MsgPort *inputDevPort,*timerport;
struct IOStdReq *inputRequestBlock;
struct Interrupt handlerStuff;
struct MemEntry me[10];
struct timerequest *timerreq;
struct IntuitionBase *IntuitionBase;
struct GfxBase *GfxBase;
struct Screen *darkscreen;

struct NewScreen dark =
 {0,0,320,30,1,0,1,NULL,CUSTOMSCREEN,NULL,NULL,NULL,NULL};

LONG ereigniss;
ULONG sekunden;
BOOL Bild=TRUE;

void WaitForTimer(tr,seconds)
struct timerequest *tr;
ULONG seconds;
{
   tr->tr_node.io_Command=TR_ADDREQUEST;
   tr->tr_time.tv_secs=seconds;
   tr->tr_time.tv_micro=0;
   DoIO(tr);
}

void dunkel()
{
   if((darkscreen=
  (struct Screen *)OpenScreen(&dark))!=0)
     {
       OFF_DISPLAY;
       SetRGB4(&(darkscreen->ViewPort),0,0,0,0);
       Bild=FALSE;
     }
}

void hell()
{
   if(darkscreen!=0)CloseScreen(darkscreen);
   ON_DISPLAY;
   Bild=TRUE;
}

void _main()
{
   if((IntuitionBase=(struct IntuitionBase *)
     OpenLibrary("intuition.library",0))==NULL)
     XCEXIT(10);
   if((GfxBase=(struct GfxBase *)
     OpenLibrary("graphics.library",0))==NULL)
     goto abbruch6;

   if((inputDevPort = CreatePort(0,0))==NULL)
    goto abbruch5;
   if((inputRequestBlock =
   CreateStdIO(inputDevPort))==NULL)
     goto abbruch4;
```

```
    if(OpenDevice("input.device",0,inputRequestBlock,0)!=0)
      goto abbruch3;

    if((timerport=CreatePort(0,0))==NULL)goto abbruch2
  if((timerreq=(struct timerequest *)
      AllocMem(sizeof(struct timerequest),
      MEMF_PUBLIC))==NULL)
       goto abbruch1;
    timerreq->tr_node.io_Message.mn_Node.ln_Type=
      NT_MESSAGE;
  timerreq->tr_node.io_Message.mn_Node.ln_Pri=0;
  timerreq->tr_node.io_Message.mn_ReplyPort=timerport;
  if(OpenDevice(TIMERNAME,UNIT_VBLANK,timerreq,0)!=0)
     goto abbruch0;

  handlerStuff.is_Data = (APTR)&me[0];
  handlerStuff.is_Code = HandlerInterface;
  handlerStuff.is_Node.ln_Pri = 51;
  inputRequestBlock->io_Command = IND_ADDHANDLER;
  inputRequestBlock->io_Data = (APTR)&handlerStuff;
  DoIO(inputRequestBlock);

  while(!CONTROL_C)
     {
      WaitForTimer(timerreq,TIMEINTERVAL);
       if(ereigniss==0)
          {
           if(Bild && (++sekunden > LIMIT))
              dunkel();
          }
       else
          {
            ereigniss=0;
            sekunden=0;
            if(!Bild)hell();
          }
     }

   inputRequestBlock->io_Command = IND_REMHANDLER;
   inputRequestBlock->io_Data = (APTR)&handlerStuff;
       DoIO(inputRequestBlock);

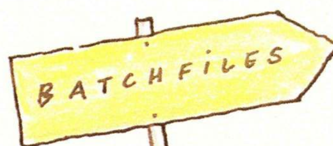
       CloseDevice(timerreq);
  abbruch0:
       FreeMem(timerreq,sizeof(struct timerequest));
 abbruch1:
     DeletePort(timerport);
 abbruch2:
     CloseDevice(inputRequestBlock);
 abbruch3:
     DeleteStdIO(inputRequestBlock);
 abbruch4:
     DeletePort(inputDevPort);
 abbruch5:
     CloseLibrary(GfxBase);
 abbruch6:
     CloseLibrary(IntuitionBase);
      XCEXIT(20);
 }
```

VON ANDREAS DIEBOLD

JUMP

Neues vom CLI





Was ist Jump? Falsch! Jump hat nichts mit Van Halen zu tun. Nein, es ist auch kein militärischer Befehl zum Eintauchen des Körpers in feuchte, wohltuende Schlammasse und hat auch ganz und gar nichts mit Skigymnastik zu tun. Es ist schlicht und einfach ein neuer Befehl für unser geliebtes CLI.

Das CLI hat, wie aus dem CLI-Kurs zu entnehmen war, bekanntlich viele Kniffe und Geheimnisse. Aber irgendwann ist einmal Schluß. 'JUMP' ist keiner dieser geheimnisvollen Befehle; es ist ein eigenständiges Programm, das sich aber im Endeffekt wie ein ganz normales CLI-Kommando verhält. Ein Blick in den System-Ordner 'c' verrät, daß es sich auch bei den normalen CLI-Befehlen um eigenständige Programme handelt. Also was liegt näher, als dem noch eines anzufügen.

Das Menü im CLI

JUMP ermöglicht benutzerkontrollierte Sprünge in beliebigen Batchfiles des AmigaDos. Am interessantesten dürfte dies in der StartupSequence sein. Ein Beispiel hierfür ist in Bild 1 zu sehen. Nach Booten des Rechners fragt dieser nun ab, ob man ins CLI, ins AmigaBasic oder die Workbench laden will.

Je nach Tastendruck verzweigt das Programm dann auf ein entsprechen-

des Label. Dieses darf ein Zeichen lang sein und muß der Taste entsprechen, die der Benutzer drückt, wenn JUMP ihn dazu auffordert.
Der Aufruf lautet folgendermaßen:

```
JUMP string [beliebige Anzahl möglicher Labels]
```

<string> ist eine Zeichenkette, die dem Benutzer angezeigt wird sobald ein Batchfile das JUMP-Kommando aufruft.
Nachdem das Programm den String auf dem Bildschirm angezeigt hat, wartet JUMP auf einen Tastendruck des Benutzers. Ist dies geschehen, springt es logischerweise zu dem Label, das den gleichen Namen hat wie die gedrückte Taste.
Intern realisiert JUMP diesen Sprung unter Zuhilfenhame des DOS-Kommandos <SKIP>. Das Label wird, wie sollte es auch anders sein, mit dem DOS-Kommando 'LAB' gekennzeichnet.

DO IT WITH...

Vorsorge ist bekanntlich besser als unangenehme Folgen. Deshalb prüft JUMP nach, ob die angegebenen Sprünge überhaupt möglich sind. Sollte es nachlässigerweise vorkommen, daß ein Label nicht existiert (Schlamperei), so wird das Batchfile mit einer Fehlermeldung abgebrochen. Weiterhin beschränkt JUMP die zulässigen Tasteneingaben auf die im Menü verlangten (im Beispiel 'c' 'w' 'a'). Drückt man also eine andere Taste, versucht JUMP nicht, in die tiefen Abgründe der Rechnerwelt zu stürzen, sondern fordert kühl und unüberwindbar den Benutzer zur erneuten Eingabe auf.

JUMP IN AKTION

Ein einfaches, oben schon angesprochenes Beispiel des Programmes kommt in der Startup-sequence zur Gel-tung. Es sieht folgendermaßen aus:

Bild 1: Ein kleines Beispiel mit großer Wirkung.

```
setmap d

JUMP "c=CLI w=WORKBENCH a=AMIGABASIC" c w a

LAB a
run amigabasic
quit

LAB c
if exists sys:system
  path sys:system add
endif
info
quit

LAB wb
loadwb
```

Anhand dieses Beispiels dürfte es keine Schwierigkeiten mehr machen, sich eigene Menüs zu generieren. Als Anregung könnte man folgende Ideen (zu sehen auf der nächsten Sei-te)

Jump erlaubt benutzergesteuerte Sprünge

```
STRPTR comm,arg[3],cp;
struct IntuitionBase *IntuitionBase;
struct Window *w;
struct MsgPort *wpBuf,*upBuf;
struct IntuiMessage *mess;

UBYTE marke[2];

main(argc,argv)
  SHORT argc;
  STRPTR argv[];
   {
   SHORT i;

   if(argc < 2) /* Abfrage nach Argumentübergabe */
     {
     Write(Output(),"Usage : jump \" <string> \" ",26L);
     Write(Output(),"[x y z ...] (=gültige Antw.)\r\n",34L);
     ende(100L);
     }
   Write(Output(),"\r\n",2L);
   Write(Output(),argv[1],(LONG)strlen(argv[1]));
   Write(Output(),"\r\n",2L);

   IntuitionBase = (struct IntuitionBase *)OpenLibrary("intuiti
         on.library",0L);
   if(!IntuitionBase)ende(100L);

/****************************************************
   * Den Zeiger auf das CLI-Window, bekommt das Programm
   * durch Abfrage des aktiven Windows. Daher ist es
   * nötig das CLI-Window während des Programmstartes
   * aktiv zu lassen.
   ***************************************************/
   w = IntuitionBase->ActiveWindow;

/****************************************************
   * Das Öffnen von IDCMP auf das CLI-Window, hat den
   * Vorteil daß nach einem Tastendruck kein
   * zusätzliches <RETURN> erfolgen muß. Damit nach dem
   * Programm die Konsole wieder zur Verfügung steht,
   * müssen allerdings die Ports des Windows gerettet
   * werden, bevor IDCMP benutzt wird.
   ***************************************************/
   wpBuf = w->WindowPort;
   upBuf = w->UserPort;
   ModifyIDCMP(w,VANILLAKEY);

   FOREVER
     {
```

```
    /* Warten auf einen Tastendruck */
    Wait(1L<<w->UserPort->mp_SigBit);

    while(mess = (struct IntuiMessage *)
                  GetMsg(w->UserPort))
     {
     /*********************************************
       * Der Asciiwert der gedrückten Taste, findet *
       * sich in den niedrigeren 8-Bit der 16-Bit   *
       * Variablen <mess->Code> wieder.             *
       * cp ist ein Zeiger auf 8-Bit der auf dieses *
       * 16-Bit Wort zeigt. Wird cp um eins         *
       * hochgezählt, so zeigt es auf den Asciiwert *
       * der gedrückten Taste.                      *
      *********************************************/
      cp = (STRPTR)&mess->Code;
      ReplyMsg(mess);

     /*********************************************
       * Wenn im Programmaufruf bestimmte Antworten *
       * vorgegeben werden, so wird die gedrückte   *
       * Taste hier mit diesen verglichen.          *
      *********************************************/
       if(argc > 2)
        {
         for(i = 2;i < (argc + 1) ; i++)
          {
           if( *argv[i] == *(cp+1) )
            {
             argc = 2;
             break;
            }
          }
        }

     /*********************************************
       * Ist die Antwort erlaubt, so wird hier der  *
       * Sprung vorbereitet und ausgeführt.         *
      *********************************************/
       if(argc == 2)
        {
         marke[1] = 0;
         marke[0] = *(cp+1);
         arg[1] = &marke[0];
         arg[2] = NULL;

         fexecv("skip",arg);
         ende(NULL);
        }

      /* Falls nicht ... */
     Write(Output(),"So geht's nicht !! Try it again\n",32L);
     } /* end of while */
   } /* end of FOREVER */
 } /* end of main */

/*****************************************************
 * Das Aufräumen :
 * Die IDCMP werden wieder entfernt, und dem Window
 * die Ports der Konsole zurückgegeben.
 *****************************************************/
ende(retval)
 LONG retval;
 {
  if(IntuitionBase)
   {
    ModifyIDCMP(w,NULL);
     w->UserPort = upBuf;
     w->WindowPort = wpBuf;
    CloseLibrary(IntuitionBase);
   }
  Exit(retval);
 }
```

verwirklichen:

- Aufruf des bevorzugten Text/ Zeichen/Terminalprogramms
- Formatieren einer Diskette
- Einstellen verschiedener Tastatur-belegungen
- Kopieren bestimmter Files nach RAM:
- Directorywahl
- Farbwahl des Bildschirms
- und ca. 387 andere Möglichkeiten

Natürlich kann jede Boot-Diskette ein anderes, speziell auf eine Anwendung zugeschnittenes Menü, enthalten. Man denke nur an eine Bilder-Show oder eine Utility- bzw. Kopierdiskette. Auch der Aufbau einer Shell für Programmiersprachen wäre möglich.

Welcher Compiler ?

JUMP wurde mit dem Aztec C 3.4a erstellt. Die Optionen hierfür lauten:

```
cc jump.c
ln jump.o -lc
```

Anschließend ist der neue Befehl fertig und wartet auf seinen Einsatz. Aber halt - zuvor muß er natürlich in den 'c'-Ordner der Boot-Diskette kopiert werden, sonst war das Tippen vergebens. Das ist aber kein Problem, nur vergessen darf man es eben nicht (bei manch defektem Rechner war bekanntlich nur der Stecker nicht in der Dose).

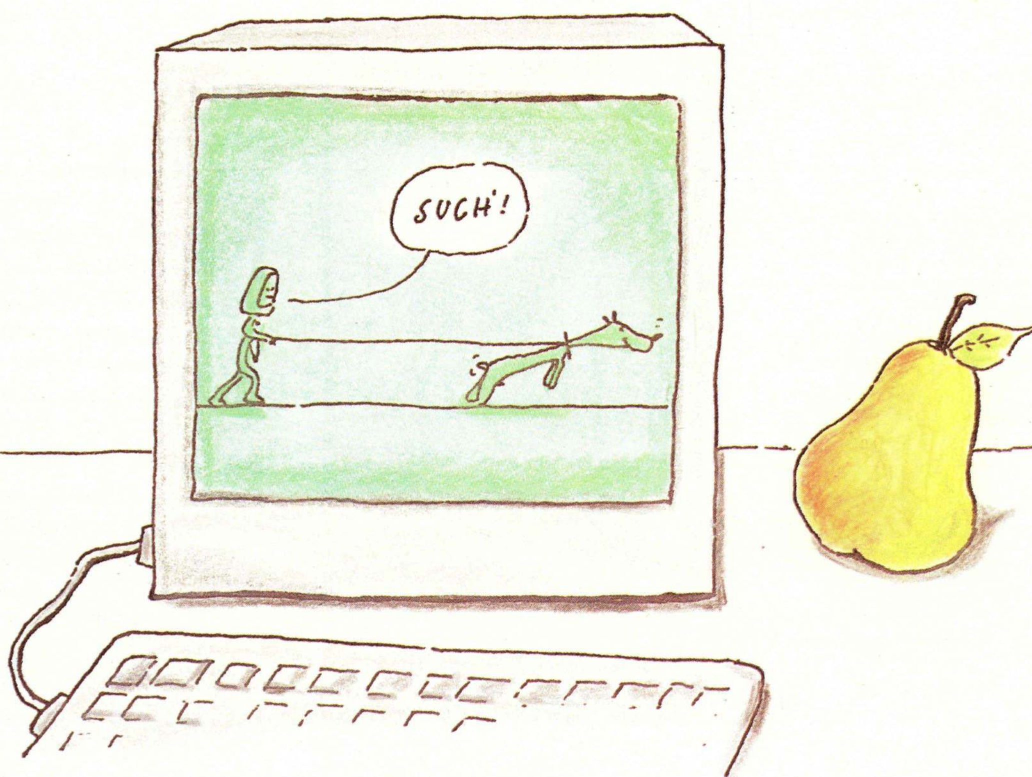
Platz ist kostbar

Die im Programm verwendete Ausgabe-Funktion 'Write()' verkürzt die Programmlänge gegenüber der Standard-Funktion 'printf()' von 7568 auf 3668 Bytes (Gruß an Kernighan & Ritchie). Diesen Platz läßt man sich natürlich nur ungern nehmen.
Nun aber viel Spaß beim JUMPen.

VON ROBERT MALZAN

WO WAR'S DOCH GLEICH?

Find hilft Ihnen weiter



Wer sich schon öfter durch den INCLUDE-Dschungel einer Compilerdiskette gewühlt hat, weiß, was ich meine: Man sucht irgendein soundso.h und kann sich beim besten Willen nicht erinnern, in welchem Inhaltsverzeichnis es steht. Oder: Man hat sich endlich aufgerafft, Ordnung in seine Disketten gebracht und für alles Mögliche ein extra Inhaltsverzeichnis erzeugt. Wie groß ist der Ärger, wenn man plötzlich nichts mehr wiederfindet.

Stolze Besitzer einer Festplatte haben bestimmt noch Schlimmeres erlebt, aber mir hat es auch so schon gereicht. Also bin ich zur Tat geschritten. Das Ergebnis ist eine nette kleine Utility, die unter Aztec-C nicht mehr als etwa drei Kbytes an Kode ausmacht. Zweck des Programms ist, alle Inhaltsverzeichnisse nach einer bestimmten Datei zu durchsuchen, ähnlich wie das allseits bekannte SEARCH alle Dateien nach einer bestimmten Zeichenkette durchsucht. Um diese Verwandschaft anzudeuten, habe ich es FIND genannt. Wer sich den Aufruf einfach machen

möchte, tippt einfach

'FIND DF1: Dateiname'

und das Programm sucht schon los, wobei es laufend Auskunft darüber gibt, welches Verzeichnis gerade durchsucht wird. Wenn nur der Dateiname angegeben wird, sucht FIND immer ab ':', also ab der Wurzel des aktuellen Verzeichnisses. Selbstverständlich kann das Laufwerk beliebig angegeben werden - df0: oder dh0: bzw. jh0:. Auf den Luxus der Joker wollte ich natürlich auch nicht ganz verzichten, aber die vom AmigaDOS leider nicht unterstützten '?' und '#' waren mir doch zu aufwendig. Deshalb habe ich nur einen '*' vorgesehen, der den meisten Bedürfnissen gerecht wird und genau wie die Kombination '#?' funktioniert.

Aber jetzt ist es Zeit für ein paar klärende Worte zum Listing. Fast alle verwendeten Funktionen sind DOS-Befehle, die speziell für die Verzeichnisstruktur angeboten werden. Um diese Funktionen zu verstehen, muß man sich zunächst mit den obligatorischen .h-Files beschäftigen, in denen die Strukturen 'FileLock' und 'FileInfoBlock' beschrieben sind (dos.h, dosextens.h). Der Befehl 'Lock()' liefert einen Pointer auf eine Struktur vom Typ FileLock, wenn der angegebene Name existiert, sonst ist das Ergebnis NULL. Der Befehl kann verwendet werden, um zu testen, ob eine Datei existiert, der zurückgelieferte Pointer

```
AmigaDOS
1>
1>
1> find df1: custom.h
Searching df1:Lib
Searching df1:shell
Searching df1:CompactH
Searching df1:CompactH/clib
Searching df1:CompactH/libraries
Searching df1:CompactH/workbench
Searching df1:CompactH/exec
Searching df1:CompactH/graphics
Searching df1:CompactH/intuition
Searching df1:CompactH/resources
Searching df1:CompactH/hardware
Found
df1:CompactH/hardware/custom.h
Searching df1:CompactH/devices
Searching df1:CompactH/proto

Searching df1:Examples

1> 1> 1> 1> 1> 1> 1> 1> 1> 1> 1> 1> 1> 1> 1> 1> 1> preferences
1> █
```

wird aber auch beim Aufruf anderer DOS-Befehle als Parameter verwendet, denn er repräsentiert das Inhaltsverzeichnis, auf das gerade zugegriffen wird. Der FileInfoBlock wird durch die Funktionen Examine() und ExNext() ausgefüllt und steht für einen Eintrag im angegebenen Verzeichnis. Dort steht alles Wissenswerte über eine Datei, wie zum Beispiel ihre Größe, ihr Erzeugungsdatum oder ihr Typ. Der Typ, genauergesagt fib_DirEntryType ist größer Null, wenn der Eintrag der Name eines weiteren Verzeichnisses ist. Wenn ich auf einen solchen Eintrag stoße, durchsuche ich immer erst dieses Verzeichnis, bevor ich weiter im aktuellen Verzeichnis suche. Dieses 'Suchen in der Tiefe' ist praktischer, als das aktuelle Verzeichnis zuerst nach Dateien und anschliessend nach Unterinhaltsverzeichnissen zu durchforsten, weil ich ja sonst zweimal dasselbe Verzeichnis lesen müsste, was bei den umständlichen AmigaDos-Zugriffen unnötig Zeit kosten würde. Das Vorgehen ist also: Lock() aufrufen, um Zugriff auf das gewünschte Verzeichnis zu haben, Examine(), um eine FileInfoBlock-Struktur für dieses Lock zu initialisieren, und in einer Schleife mit ExNext() einen Eintrag nach dem anderen abzuarbeiten, bis ich keinen mehr finde (ExNext()==FALSE).

Wer sich den Sourcecode von FIND schon angesehen hat, wird bemerkt haben, daß sich die Prozedur FFind() selbst aufruft, wenn ein Unterinhaltsverzeichnis auftaucht. Solche Funktionen heißen 'rekursiv' - lateinisch für 'zurückkehren' -, weil sie (sofern der Programmierer nicht gepfuscht hat), immer auf dem 'Weg' zurückkehren, den sie beim sich-selbst-Aufrufen genommen haben. Man kann sich das auch weniger kompliziert vorstellen: Angenommen, man hätte beliebig viele Kopien ein und derselben Funktion. Wenn nun die erste Funktion eine Leistung benötigt, wie sie von dieser Funktion selbst geleistet wird, dann ruft sie die zweite auf, welche bei Bedarf die dritte aufruft... usw. Bei dieser Art der Programmierung spielt der Unterschied zwischen lokalen und globalen Variablen eine große Rolle. In der Funktion FFind() gibt es ein char-A-ray namens Path[]. Da das Array lokal definiert ist, existiert es nur für diese eine Funktion (ich beziehe mich jetzt auf das Modell mit den unendlich vielen Kopien einer Funktion) und wird bei Verlassen derselben wieder 'vergessen'. Diese Wirkung mache ich mir für meinen aktuellen Pfadnamen zunutze, der sich ja verlängert, wenn ich ein Unterinhaltsverzeichnis durchsuche, aber erhalten bleiben muß, um im aktuellen Inhaltsverzeichnis weitersuchen zu können. So macht sich jede 'Version' von FFind() eine lokale Kopie vom gerade aktuellen Pfadnamen, der behalten wird, solange diese 'Version' nicht 'ihr' Inhaltsverzeichnis komplett abgearbeitet hat. Bei der Variablen Code hingegen, die ich zum Abbrechen der Suche verwende (mit CTRL-C), war es im Gegenteil unerläßlich, sie als global zu deklarieren, weil man sonst bei einer Funktion, die sich selbst zehnmal aufgerufen hat, auch zehnmal CTRL-C drücken müßte, um abzubrechen - ein mühsames Unterfangen! Das übrige Programm dürfte selbsterklärend sein. Es wurde unter Aztec-C mit

cc Find -IINCLUDE:

und

ln Find.o -Lc

erzeugt, wobei INCLUDE: der Pfadname für die include-Dateien ist. Man könnte das Programm sicher noch kompakter schreiben, aber ich habe

bewußt auf die in C so verführerischen Kurzschreibweisen verzichtet, weil das für die Lesbarkeit des Programms nicht gerade förderlich wäre. Ich wünsche gutes ~~Vor~~ankommen mit FIND und gute Ideen, die den AMIGA mal so richtig aus der Reserve locken...

```
/*------------------------------
        FIND
 Autor: Bob Malzan
 (c) KICKSTART Magazin 1988
 FIND sucht eine (mit Joker) spezifizierte
 Datei in allen Unterinhaltsverzeichnissen
 und gibt, falls gefunden, den dazugehoerigen
 Pfadnamen aus.
 ------------------------------*/


#include <libraries/dos.h>
#include <libraries/dosextens.h>
#include <exec/memory.h>


APTR        DosBase,OpenLibrary(),AllocMem();
struct File      *Out,*Output();
struct FileLock *Lock();
VOID              Unlock(),FreeMem();
char              toupper(),*strcat();
char              *LockName,*SearchName;
BOOL              EntryFound,Examine(),ExNext();
BOOL              strcmp();
BOOL              Found=FALSE;
LONG              Code=0L;

char ClrEoln[] = { 0x1B, 0x9B, 'K' };
#define CLREOLN Write(Out,ClrEoln,3L)

/*--- Hilfsroutinen ----------*/

/* PrintOut loescht die aktuelle Zeile und *
 * schreibt String nach Out                */

VOID PrintOut(String)
char *String;
{
   CLREOLN;
  Write(Out,String,(long)strlen(String));
}

/* StrToUpper wandelt einen String in einen *
 * UpperCase-String                         */

VOID StrToUpper(Str)
char *Str;
{
  while(*Str!=(char)0) {
    *Str=toupper(*Str);
    Str++;
  }
}

/* StrEqual vergleicht zwei Strings, wobei *
 * Str1 '*' als Joker enthalten darf.      */

BOOL StrEqual(Str1,Str2)
char *Str1,*Str2;
{
  UBYTE P1=0,P2=0;

  while((strlen(Str1)>P1)&&(strlen(Str2)>P2)){
    if(*(Str1+P1) != toupper(*(Str2+P2))) {
      if(*(Str1+P1)=='*') {
        if(strlen(Str1)==P1+1) return(TRUE);
        P1++;
        while ( (strlen(Str2)>P2) && (toupper
          (*(Str2+P2)) != *(Str1+P1)) ) P2++;
        if ( strlen(Str2)==P2 ) return(FALSE);
      }
      else return(FALSE);
    }
    P1++; P2++;
  }
  if((*(Str1+P1)=='*')&&(P1+1==strlen(Str1)))
   return(TRUE);
  if((P2<strlen(Str2))||(P1<strlen(Str1)))
   return(FALSE);
  else return(TRUE);
}

/* FFind - die rekursive Prozedur, die alle *
 * Unterinhaltsverzeichnisse durchlaeuft.  */

VOID FFind(LName,First)
char *LName;
BOOL First;
{
  struct FileLock *CurrLock;
  char Path[80];
 /* ist ein lokaler String, in dem sich *
  * der Pfadname befindet              */
  struct FileInfoBlock *FInfo;

  CurrLock=Lock(LName,SHARED_LOCK);
  if (CurrLock!=NULL) {
    FInfo=(struct FileInfoBlock *)
     AllocMem((LONG)sizeof(struct FileInfoBlock)
                                ,MEMF_PUBLIC);
    if (FInfo==NULL) {
      PrintOut("Memory Problems!!\n");
      UnLock(CurrLock);
      return;
    }

    EntryFound=Examine(CurrLock,FInfo);
    EntryFound=ExNext(CurrLock,FInfo);
    while ((EntryFound) &&
                !(Code&SIGBREAKF_CTRL_C)) {
      Code=SetSignal(0L,SIGBREAKF_CTRL_C);
      strcpy(&Path[0],LName);
      if (StrEqual(SearchName,&FInfo->
                          fib_FileName)) {
        PrintOut("Found ");
        PrintOut(&Path[0]);
        if (!First) PrintOut("/");
        PrintOut(&FInfo->fib_FileName);
        if (FInfo->fib_DirEntryType>0)
          PrintOut(" <DIR>");
        PrintOut("\n");
        Found=TRUE;
      }
      if (FInfo->fib_DirEntryType>0) {
        if (!First) strcat(&Path[0],"/");
        strcat(&Path[0],&FInfo->
                    fib_FileName[0]);
        PrintOut("Searching ");
        PrintOut(&Path[0]);
        PrintOut("                    \r");
        FFind(&Path[0],FALSE);
      }
      EntryFound=ExNext(CurrLock,FInfo);
    }
    FreeMem(FInfo,(LONG)sizeof(struct
                        FileInfoBlock));
    UnLock(CurrLock);
  }
}

main(argc,argv)
int argc;
char *argv[];
{
  ULONG Length;
  BOOL First=FALSE;
  char Test;

  DosBase=OpenLibrary(DOSNAME,NULL);
  Out=Output();
  if (Out==NULL) exit(20L);
  argc-;
  if((argc<1)||((argc==1)&&(**(argv+1)=='?')))
   {
    PrintOut("Usage: ");
    PrintOut(*argv);
    PrintOut(" [ROOT],NAME/A\n");
    exit(0L);
   }
```

```
    if (argc==1) {
      LockName=":";
/* no ROOT specified -> current device root */
      SearchName=*(argv+1);
    }
    else {
      LockName=*(argv+1);
      SearchName=*(argv+2);
    }

    StrToUpper(SearchName);
    Test=*(LockName+strlen(LockName)-1);
    if ( (Test==':')||(Test=='/') ) First=TRUE;
    FFind(LockName,First);
    if (Code&SIGBREAKF_CTRL_C) {
      PrintOut(*argv);
      PrintOut(" aborted by ^C");
    }
    else if (!Found) {
      PrintOut("Couldn't find ");
      PrintOut(SearchName);
    }
    PrintOut("                    \n");
    CloseLibrary(DosBase);
} /* Programm Ende */
```

'FIND' hilft bei der sonst so aussichtslosen Suche nach verschollenen Files.

WE WANT YOU!

Wie schon auf der Titelseite erwähnt, wird diese Rubrik zu einem ständigen und wichtigen Bestandteil dieser Zeitschrift. Gedacht ist sie für all diejenigen, die ihren Rechner selbst programmieren, und dazu Tips und Anregung gebrauchen können. Diese Rubrik kann aber nur dann bestehen, solange viele Leser (Sie eingeschlossen) sich daran beteiligen. Wir fordern Sie deshalb auf, Ihre Ideen in einen Umschlag zu stecken und auf dem schnellsten Weg zu uns zu schicken. Vorraussetzungen gibt es praktisch keine: es muß lediglich interessant sein und so dokumentiert, daß auch andere User daraus einen Nutzen ziehen können. Auch bei den Sprachen gibt es keinerlei Einschränk-ungen. Ob C, BASIC, PASCAL, Assembler, Modula-2, Fortran, Forth, LISP oder andere Sprachen bleibt Ihnen überlassen. Veröffentlichte Programme werden natürlich angemessen honoriert.

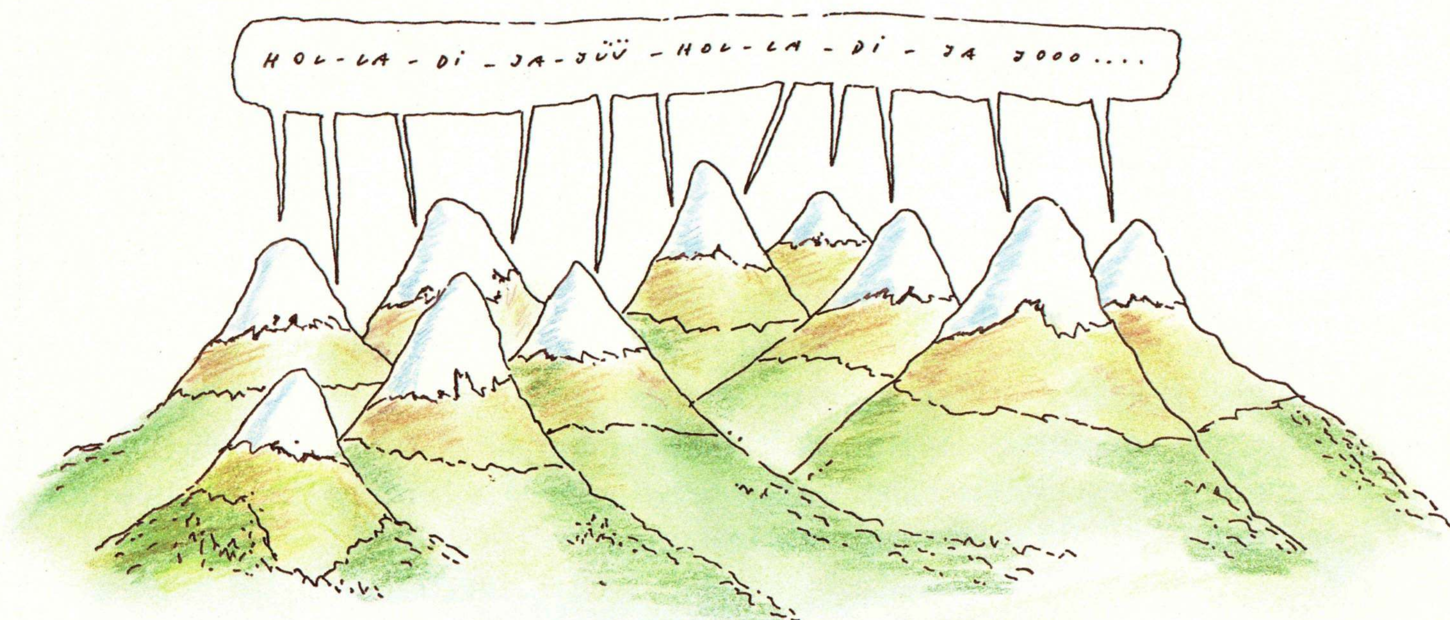
Bitte beachten Sie bei der Einsendung: Schicken Sie den Quelltext und das evtl. kompilierte Programm sowie die Dokumentation auf Papier und Diskette. Die Diskette wird zurückerstattet. Einsendungen direkt an die Redaktion:

MERLIN-Computer GmbH, Redaktion KICKSTART 'KICKS'
Industriestr. 26, D-6236 Eschborn, Tel: 06196 / 481811

VON MICHAEL BRINKMANN

AUFGEHT´S

Fraktale Landschaften in Basic



Das Titelblatt der Juli/August-Ausgabe von Kickstart klang verheißungsvoll: 'Listings: 3D-Fractals'. Diese Art der Grafik hatte mich schon immer fasziniert, doch bisher war es mir nicht gelungen ein Programm zu schreiben, das fraktale Grafiken erstellt. Ich blätterte also voller Freude zur Seite 66, doch dann wurde ich schwer enttäuscht:

Das Listing war in C geschrieben. Da ich der Sprache C nicht mächtig bin, konnte ich aus dem Programmtext nicht erfahren, wie die Grafik berechnet wird. Trotzdem beschloß ich, ein Fractalprogramm zu schreiben, und zwar eines, das jeder benutzen kann. So entstand das hier gedruckte Listing in Amiga-Basic. Das Berechnen und Zeichnen der Grafik nimmt nur 75 Sekunden in Anspruch.

Zur Arbeitsweise

Nach dem Start des Programms öffnet sich zuerst ein Screen und dann ein Fenster. Danach wird nach einer Zahl zwischen -32768 und +32767 gefragt. Diese Zahl dient als Startwert für den Zufallszahlengenerator und bestimmt die Anfangsgrafik. Nach einigen Farb- und Variablendeklarationen wird die erste Grafik berechnet. Nachdem die Nachricht 'Taste drücken' auf dem Bildschirm erschienen ist, wird nach Druck auf eine beliebige Taste die Landschaftsgrafik dargestellt. Während diese Landschaft noch zur Betrachtung steht, erfolgt bereits die nächste Berechnung.

Nachdem wir die grundlegenden Dinge geklärt haben, wollen wir auf die Berechnung der Landschaften eingehen. Wie man sehen kann besteht die Landschaft aus vielen Dreiecken. Diese Dreiecke sind aber nichts anderes als mit Linien verbundene Punkte. Jetzt gilt es nur noch die Höhe dieser Punkte zu berechnen. Um die Berechnung etwas zu vereinfachen, gehen wir ersteinmal von der 3D- in die 2D-Ebene und nehmen uns hier 17 nebeneinanderliegende Punkte. Punkt 1 und Punkt 17 werden auf einen zufälligen Wert zwischen -8 und +8 gesetzt. Die beiden Werte sind dabei voneinander unabhängig. Verbindet man nun P1 mit P17, ergibt sich eine Gerade, auf der die Punkte P2-P16 liegen, von denen jeder eine bestimmte Höhe hat (Bild 1). Wir halbieren nun die Gerade im Punkt 9. Die Höhe dieses Punktes wird aus der Hälfte der Höhe von P1 und P17 berechnet.

```
Höhe(P9)=(Höhe(P1)+Höhe(P17))/2
```

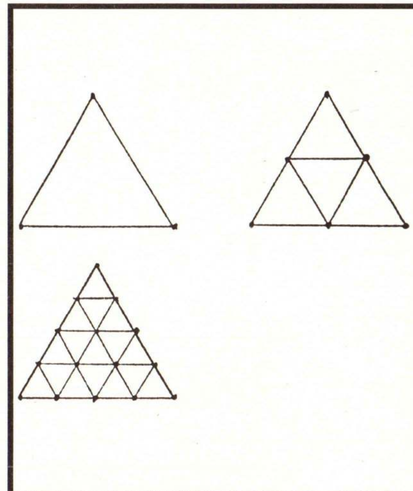
Zu dieser berechneten Höhe addieren wir einen zufälligen Wert von -8 bis +8 . Nach dem Verbinden von P1 mit P9 und P9 mit P17 ergeben sich zwei Geraden, die abermals halbiert werden. Die Höhe der Punkte P5 und P13 läßt sich nach der oben genannten Formel berechnen, wobei natürlich P1 durch P9 und P17 durch P9 ersetzt gehört. Zu der Höhe addiert man dann noch einen Zufallswert von -4 bis +4, also die Hälfte des vorherigen Zufallswertes. Die vier entstandenen Geraden halbiert man wieder und addiert den Zufallswert von -2 bis +2 (wieder die Hälfte des vorherigen). Dies ergibt acht Geraden, mit denen genauso verfahren wird, nur mit einem Zufallswert von -1 bis +1.

Wir haben nun 17 Punkte mit unterschiedlichen Höhen. Der Höhenunterschied zwischen zwei nebeneinanderliegenden Punkten kann maximal 5 betragen. Außerdem ist es nicht möglich, daß P1 die Höhe Null hat, P2 dann 5 und P3 wieder Null, da ein Punkt von seinen Nachbarn abhängig ist.

Jetzt muß nur noch das Modell in die 3D-'Ebene' umgesetzt werden. Dazu nehmen wir das Dreieck, dessen Eckpunkte auf einen beliebigen Wert gesetzt werden müssen. Nachdem man die Eckpunkte miteinander verbunden hat, ergeben sich drei lange Geraden, die in drei unterschiedlichen Richtungen verlaufen. Die im Laufe der Berechnungen entstandenen Geraden werden immer in einer der drei Richtungen verlaufen. Halbiert man zwei der langen Geraden und verbindet deren Mittelpunkte miteinander, ergibt sich eine neue, kürzere Gerade. Mit dieser kürzeren Gerade muß man genauso verfahren wie mit den im vorangegangenen Text beschriebenen. Teilt man die langen Geraden noch einmal, entstehen zwei weitere Geraden. Es kommt noch die Teilung der dritten langen Geraden hinzu, die zur Folge hat, daß noch weitere Geraden in den anderen beiden Richtungen entstehen, wenn man die Punkte miteinander verbindet (Bild). Bei jeder Teilung der langen Geraden muß auch eine Teilung der kürzeren erfolgen, und außerdem muß der Zufallswert halbiert werden. Ist schließlich jedem Punkt eine Höhe zugewiesen, stellt man fest, daß jeder Punkt von seinen umliegenden abhängig ist. Daraus ergibt sich dann eine Landschaftsform, die man mit einer geeigneten Zeichenroutine darstellen kann.

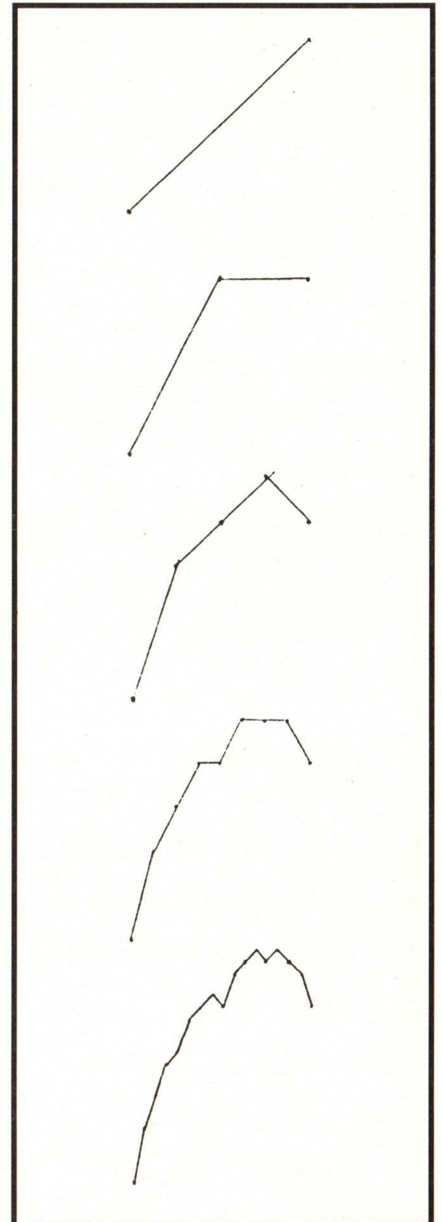
Die Anzahl der Punkte, aus denen die äußeren Geraden bestehen, werden noch von 17 auf 65 hochgesetzt, wodurch sich eine bessere Auflösung und so ein besseres Bild ergibt. Aufgrund der höheren Auflösung muß aber auch der Startzufallswert erhöht werden, in unserem Fall von acht auf 32. Erhöht man den Wert noch mehr, kommen bei den Bergen größere Steigungen heraus.

Ich hoffe , ich habe Sie dem Prinzip der fraktalen Grafik etwas näher gebracht, und wünsche Ihnen viel Spaß mit dem doch recht kurzen Programm.



Die Umsetzung mit Dreiecken

Die Entwicklung einer 2D-Fraktal Gerade



```
REM  Fraktale Landschaft in AmigaBasic
REM  (c) 1988 KICKSTART
REM

DEFINT a-z
DIM hoehe(66,66),farbe(401),effh(401)
SCREEN 2,640,256,2,2
WINDOW 2,,,0,2
RANDOMIZE
PALETTE 0,0,0,0
PALETTE 1,0,0,1
PALETTE 2,0,1,0
PALETTE 3,1,1,1

COLOR 1,0
CLS
LOCATE 2,21
PRINT "Fractale Grafiken von Michael Brinkmann"
LOCATE 4,33
PRINT "vom 8.12. 1987"
LINE (152,4)-(480,36),1,b
LOCATE 6,30
PRINT "Berechnungstiefe :"
COLOR 2,0

FOR count=0 TO 400
  IF count<=200 THEN effh(count)=0 ELSE effh(count)=count-200
  IF count>60 THEN farbe(count)=3 ELSE farbe(count)=2
NEXT count
farbe(0)=1
berechnen:
  COLOR 2,0
  zwert=80                                ' Startzufallswert
  zaehlw=64                               ' Zaehlweite
  schw=32                                 ' Schrittweite
 hoehe(1,1)  =200+(RND*zwert-zwert/2)     ' Höhe Eckpunkt 1
 hoehe(65,65)=200+(RND*zwert-zwert/2)     ' Höhe Eckpunkt 2
 hoehe(1,65) =200+(RND*zwert-zwert/2)     ' Höhe Eckpunkt 3
  FOR tiefe=1 TO 6                        ' Berechnungstiefe
    LOCATE 6,48
    PRINT tiefe
    FOR y=1+schw TO 65-schw STEP zaehlw
      FOR x=1 TO y-schw STEP zaehlw
        hoehe(x+schw,y+schw)=(hoehe(x,y+schw)+
                hoehe(x+zaehlw,y+schw)-zwert)/2+RND*zwert
        hoehe(x,y)=(hoehe(x,y-schw)+
                hoehe(x,y+schw)-zwert)/2+RND*zwert
        hoehe(x+schw,y)=(hoehe(x,y-schw)+
                hoehe(x+zaehlw,y+schw)-zwert)/2+RND*zwert
      NEXT x
    NEXT y
    zwert=zwert/2                         ' Zufallswert halbieren
    schw=schw/2
    zaehlw=zaehlw/2
  NEXT tiefe

LOCATE 8,34
PRINT "Taste drücken"
WHILE INKEY$=""                           ' Auf Taste warten
WEND

zeichnen:
  LINE (0,56)-(640,255),0,bf
  COLOR 0
  FOR y=64 TO 1 STEP-1
    FOR x=1 TO y
      ho1=effh(hoehe(x,y))
      ho2=effh(hoehe(x+1,y+1))
      ho3=effh(hoehe(x,y+1))
      farbe=farbe(ho1+ho2+ho3)
      kon1=315-y*4+x*8
      kon2=250-y*2
      AREA (kon1+4,kon2)
      AREA (kon1-4,kon2)
      AREA (kon1-4,kon2-ho3-2)
      AREA (4+kon1,kon2-ho2-2)
      AREAFILL
      LINE -(kon1,kon2-ho1),farbe
      LINE -(kon1-4,kon2-ho3-2),farbe
      LINE -(4+kon1,kon2-ho2-2),farbe
    NEXT x
  NEXT y
GOTO berechnen
```



ABO





ABO

Absender
(Bitte deutlich schreiben)

Vorname/Name

Straße/Nr.

PLZ/Ort

Postkarte

Bitte mit 60 Pf. frankieren

Heim-Verlag
Heidelberger Landstr. 194

6100 Darmstadt-Eberstadt
Telefon 0 61 51 / 5 60 57



Einzelheft- u.
Monatsdisketten
Bestellung





Einzelheft- u.
Disketten Service

Absender
(Bitte deutlich schreiben)

Vorname/Name

Straße/Nr.

PLZ/Ort

Postkarte

Bitte mit 60 Pf. frankieren

Heim-Verlag
Heidelberger Landstr. 194

6100 Darmstadt-Eberstadt
Telefon 0 61 51 / 5 60 57



Kleinanzeigen





Kleinanzeigen

Absender
(Bitte deutlich schreiben)

Vorname/Name

Straße/Nr.

PLZ/Ort

Postkarte

Bitte mit 60 Pf. frankieren

Heim-Verlag
Heidelberger Landstr. 194

6100 Darmstadt-Eberstadt
Telefon 0 61 51 / 5 60 57

KICK START AMIGA ZEITSCHRIFT

Abonnement

Ja, bitte senden Sie mir die Amiga-Computer Fachzeitschrift ab ______________________
für mindestens 1 Jahr (11 Hefte) zum ermäßigsten Preis von jährlich DM 70,— frei Haus.
(Ausland: Nur gegen Scheck-Voreinsendung DM 90,— Normalpost.)
Der Bezugszeitraum verlängert sich nur dann um ein Jahr, wenn nicht 6 Wochen vor Ablauf des Abonnements gekündigt wird.

Name

Vorname

Straße/Nr.

PLZ Ort

Gewünschte Zahlungsweise bitte ankreuzen

☐ Bequem und bargeldlos durch Bankeinzug

Konto-Nr. BLZ

Institut Ort

☐ Ein Verrechnungsscheck über DM______ liegt bei.

Garantie:
Diese Bestellung kann ich schriftlich innerhalb einer Woche (rechtzeitige Absendung genügt) widerrufen. Dies bestätige ich durch meine 2. Unterschrift.

Datum Unterschrift

Datum Unterschrift

KICK START AMIGA ZEITSCHRIFT

ABO

KICK START AMIGA ZEITSCHRIFT

Einzelheft- u. Monatsdisketten Bestellung

KICKSTART können Sie direkt beim HEIM-VERLAG zum Einzelheft-Preis von DM 7,– (zuzüglich Gebühr für Porto und Verpackung) nachbestellen. Bearbeitung nur gegen beigefügten Scheck über den entsprechenden Betrag (keine Überweisung).

Jan.	Febr.	März	April	Mai	Juni	Juli/Aug.	Sept.	Okt.	Nov.	Dez.	
											1987 = DM
											1988 = DM

+ Gebühr für Porto u. Verp. = DM

☐ Scheck in Höhe zus. DM liegt bei

Disketten Service

Alle Programme, die in KICKSTART veröffentlicht wurden, sind auf Disketten erhältlich. Die Disketten enthalten die Programme von jeweils 2 KICKSTART-Ausgaben. Bestellen Sie durch ankreuzen die gewünschten Disketten

Preis je Diskette 19,– DM	Juli/Aug.	Sept./Okt.	Nov./Dez.	Jan./Febr.	März/Apr.
	87	87	87	88	88

Lieferung: gegen beigefügten Scheck zuzügl. 5,– DM Versandkosten

KICK START AMIGA ZEITSCHRIFT

Einzelheft- u. Monatsdisketten Bestellung

KICK START AMIGA ZEITSCHRIFT

Kleinanzeigen-Auftrag

Bitte veröffentlichen Sie für mich folgende Kleinanzeige in der angekreuzten Rubrik

Biete an ☐ Hardware ☐ Software **Ich suche** ☐ Hardware ☐ Software ☐ **Tausch** ☐ **Kontakte** ☐ Verschiedenes ☐

30 Buchstaben je Standardzeile – incl. Satzzeichen und Wortzwischenräume.
Groß- und Kleinbuchstaben verwenden, fettgedruckte Wörter unterstreichen.

Bearbeitung **nur gegen Vorausscheck** über den entsprechenden Betrag (keine Überweisung)

☐ privat = DM 7,– je Zeile incl. MwSt.
☐ gewerblich = DM 15,– je Zeile + MwSt.
☐ Chiffregebühr = DM 10,–

☐ Scheck über DM______ ist beigefügt

Bei Angeboten: Ich bestätige, daß ich alle Rechte an den angebotenen Sachen besitze.

Datum Unterschrift

Absenderangaben auf der Rückseite nicht vergessen

KICK START AMIGA ZEITSCHRIFT

Kleinanzeigen



Kontaktkarte





Kontaktkarte

Bitte Adresse der Firma, bei der Sie Informationen, oder etwas bestellen möchten, auf der rechten Seite eintragen → → → → → → → → → →

Absender
(Bitte deutlich schreiben)

Vorname/Name

Beruf

Straße/Nr.

PLZ/Ort

Telefon Vorwahl/Rufnummer

Postkarte

Bitte freimachen

Firma

Straße/Postfach

PLZ Ort



Kurzmitteilung





Kurzmitteilung

Absender
(Bitte deutlich schreiben)

Vorname/Name

Straße/Nr.

PLZ/Ort

Telefon

Postkarte

Bitte freimachen

Merlin Computer GmbH
KICKSTART Redaktion
Industriestraße 26

6236 Eschborn



PD Bestellung





PD Bestellung

Absender
(Bitte deutlich schreiben)

Vorname/Name

Straße/Nr.

PLZ/Ort

Postkarte

Bitte freimachen

Merlin Computer GmbH
KICKSTART Redaktion
Industriestraße 26

6236 Eschborn

KICK START
AMIGA ZEITSCHRIFT

☐ Ich bitte um weitere Informationen

☐ Ich gebe folgende Bestellung auf

in Bezug auf Ihre Anzeige in Kickstart Heft ________ Seite ____

Menge	Produkt/Bestellnummer	DM	gesamt DM

Datum, Unterschrift (für Jugendliche unter 18 Jahren der Erziehungsberechtigte)



Abgesandt am:

KICK START
AMIGA ZEITSCHRIFT

Firma:

Bemerkungen:

KICK START
AMIGA ZEITSCHRIFT

Meine Meinung

Zu dem Artikel ____________ in Heft ____, Seite ____
hätte ich folgendes zu bemerken:

☐ Ich möchte Ihnen folgendes Programm zur Veröffentlichung anbieten: (Kurzbeschreibung, Sprache, Länge in Druckerseiten, GEM/TOS)

☐ Ich kann über folgendes Thema berichten: (Tips & Tricks am ST, Hardware, Software, etc.)

☐ Ich möchte gerne Autor in der ST-Computer werden. Meine Fachgebiete: (z.B. LISP, Pearl, Modula-2, Assembler ...)

☐ Ich möchte, daß folgendes Public-Domain Programm in Ihre Sammlung aufgenommen wird.

☐ Sonstiges

Bei weiteren Angaben oder Fragen wenden Sie sich bitte schriftlich oder telefonisch an die Redaktion. Tel. 0 61 96/48 18 11



KICK START
AMIGA ZEITSCHRIFT

Kurzmitteilung

KICK START
AMIGA ZEITSCHRIFT

PUBLIC DOMAIN SERVICE

Ich bestelle folgende PD-Disketten:
(Siehe PD Service in dieser Ausgabe)

Zahlung erfolgt:

☐ per Scheck

☐ per Nachnahme

Je Diskette fügen Sie bitte einen Betrag von DM 10,– bei,
für Porto und Verpackung je Sendung DM 5,– (Ausland DM 10,–)

Datum

Unterschrift



KICK START
AMIGA ZEITSCHRIFT

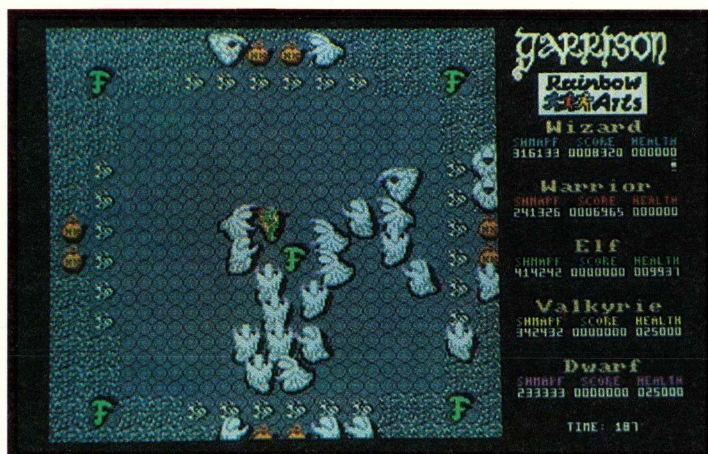
PD Bestellung

Datum

GARRISON II

The Legend continues.

Brandneu ist das Spiel Garrison II, das Programm ist der Nachfolger des erfolgreichen Spiels Garrison. Im Grunde genommen hat sich nicht viel geändert, außer daß 128 neue Levels erstellt und einige Farben der Sprites geändert wurden, doch ...



Viele Geister machen auch einem geübten Kämpfer zu

Sound und Grafik sind fast identisch mit dem Vorläufer, aber trotzdem gehört auch Garrison II wieder zu den besten Strategie-Aktion-Spielen für den AMIGA. Das Scrolling könnte etwas schneller sein, des öfteren kommt es vor, daß die Spielfigur schneller läuft als das Spielfeld scrollt.

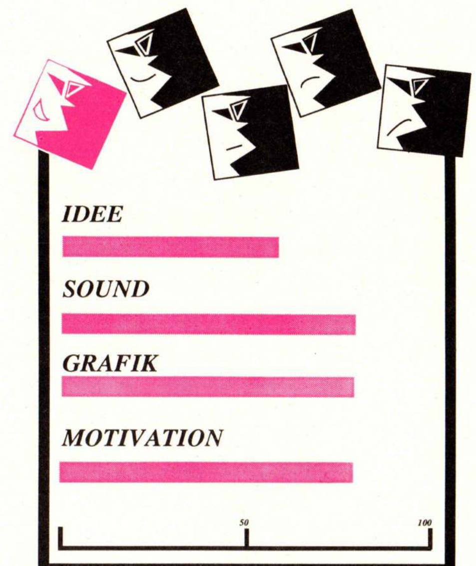
Doch zum Spiel selber. Dem Spieler stehen fünf verschiedene Charaktere zur Auswahl (Wizzard, Warrior, Elf, Valkyrie und Dwarf), wobei jede Spielfigur in allen Levels eingesetzt werden kann und Stärken und Schwächen besitzt. . Der Spieler sollte für die richtigen Levels die passenden Kämpfer aussuchen, um das Dungeaon unbeschadet zu überstehen. Die Besonderheit an Garrison war mit Sicherheit der Zweispieler-Modus, Garrison II besitzt den Modus natürlich auch. Es können hierbei zwei Spieler miteinander (nicht gegeneinander) ge-gen die Übermacht der Unterwelt an-treten. Das Spielen zu zweit ist in ei-nigen Levels durchaus sehr hilfreich: während der eine Spieler die Geister ablenkt, kann der zweite Gegenstände einsammeln oder einen geeigneten Weg suchen. In allen Levels können oder sollten verschiedene Gegenstände aufgesammelt werden, sie erleichtern des öfteren das Vorankommen ungemein.

Garrison II wird auf zwei Disketten mit einem kurzen Informationsschreiben geliefert. Ein Manko muß dem Programm aber angelastet werden. AMIGA-Besitzer mit nur 512 KByte kommen nicht in den vollen Genuß des Programms, es benötigt zum einwandfreien Ablauf 1 MByte-RAM. Besitzt der Rechner nur 512 KByte, werden alle Spielfiguren gleich dargestellt. Ein Wermutstropfen des Programms.

Fazit

Garrison II unterscheidet sich kaum vom Vorgänger. Wer Garrison I besitzt, braucht sich den Nachfolger nicht unbedingt zu kaufen, außer wenn er Interesse an 128 neuen Levels hat. AMIGA-Anwender, die noch kein Garrison besitzen, können zwischen beiden Versionen auswählen. Der Unterschied besteht nur in den verschiedenen Levels.

Anbieter: RAINBOW ARTS Software GmbH
Münsterstr. 27
4830 Gütersloh 1
Preis: 69.95 DM

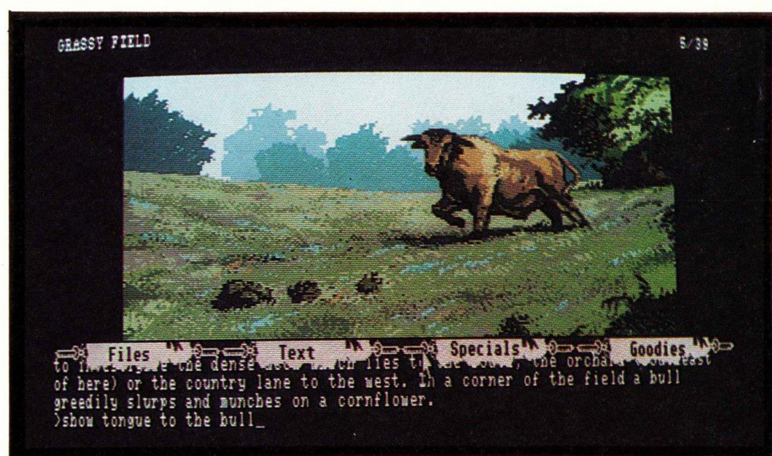


JINXTER

Slang bis zum Absturz

Ich habe einmal einen Computer und ein Spiel, das sich "Guild of thieves" nannte, in die Finger bekommen und mir diese auch zwei Nächte harter Arbeit mit meinem Freund Max - I don`t understand asshole in that context - zusammen wundgespielt.

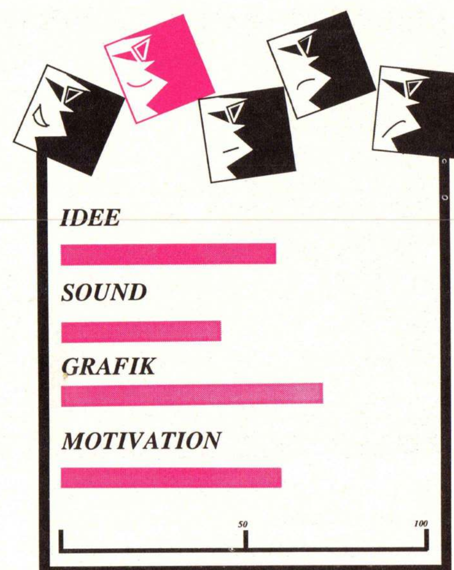
Wir waren wirklich begeistert, starben tausend Tode und freuten uns wie die Schneekönige, wenn wir dem Programm wieder einmal so schöne Formulierungen wie "demented pyromaniac" entlockt hatten. Wir wollten uns niemals wieder trennen von diesem herrlichen Zeitvertreib. Es kam aber natürlich alles ganz anders! Das Leben hatte uns bald wieder und ich wurde mit Rainbird Software bzw. Magnetic Scrolls erst wieder konfrontiert, als man mir in der Kickstart-Redaktion "JINXTER" als direkten Nachfolger der Gilde derDiebe wärmstens ans Herz legte. Ich habe in diesem Moment leider vergessen, daß ich ROCKY XXV ziemlich blöde fand, und daß ich auch den Weißen Hai nur bis Teil III aus-gehalten habe. JINXTER ist doch stark der Gilde der Diebe nachempfunden.



An diesem Tierchen muß man irgendwie vorbei.

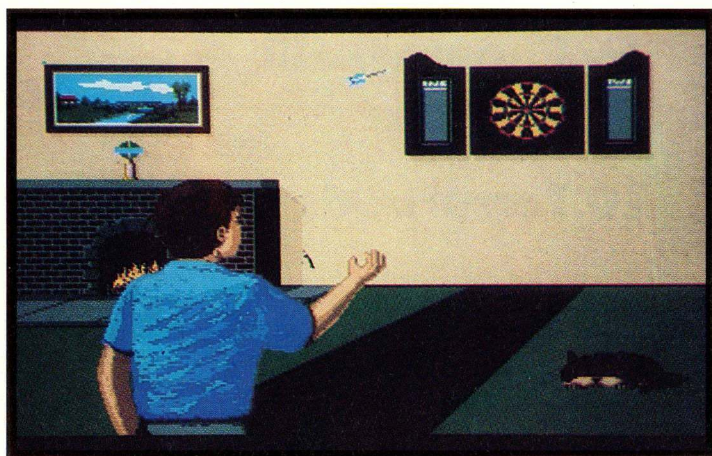
Die Geschichte der Geschichte in Aquitania (Name erfunden) hatte der Zauberer Turani (Name erfunden) einst ein besonderes Armband geschaffen. Es sollte die Bewohner durch seine magischen Kräfte vor den allgegenwärtigen bösen Hexen schützen. Dieser schönen Zeit, in der Glück und Frieden in Aquitania anzutreffen waren, wurde aber von der Hexe Jannedor (Name erfunden) und ihren Helfern ein jähes Ende bereitet, indem diese das Armband auseinandernahmen und die Einzelteile im ganzen Land versteckten. Von diesem Tage an wurde Aquitania wieder von Pech und Unglück heimgesucht. Der Spieler hat nun die ehrenvolle Aufgabe, diese Einzelteile wiederzufinden, um so dem Hexenspuk ein Ende zu bereiten. Keine leichte Aufgabe, denn JINXTER verlangt den Dialog mit dem Computer und setzt ziemlich gute Englischkenntnisse (übelster Slang) voraus. Da hilft auch die deutsche Übersetzung des "Independent Guardian", die der Spielpackung beiliegt, nicht darüber hinweg. JINXTER scheint mir schwieriger zu sein als die Gilde der Diebe, und ich gehe davon aus, daß das Bett wiedereinmal nächtelang kalt bleibt. Ran an den Käse.

Rainer Spirandelli



INDOOR SPORTS

Das Softwarehaus Mindscape, bekannt durch gute Spiele, hat mit INDOOR SPORTS seinem Ruf alle Ehre gemacht. Das Programm beinhaltet vier außergewöhnliche Sportarten - Bowling, Dartwerfen, Tischtennis und Luft-Hockey.



Zum Dartwerfen gehört äußerste Präzision.

Bis zu vier Spieler können gleichzeitig bei der Freizeit-Olympiade mitwirken, wobei jeweils zwei gegeneinander antreten. Besonders beim Tischtennis und dem Luft-Hockey macht das Spielen gegen einen Freund einen Heidenspaß. Jede Sportart wird mit dem Joystick bedient; aus diesem Grunde ist es ratsam, zwei Sticks zu besitzen, um auch gegen einen Freund antreten zu können, der ja auch einen Joystick benötigt (mit Meditation ist der AMIGA schließlich noch nicht steuerbar).

Alle Disziplinen besitzen einen besonderen Reiz, nur durch gezieltes Einsetzen des Joysticks kann ein größerer Erfolg verbucht werden. Etwas Übung ist notwendig, um alle Disziplinen zu beherrschen oder sich lange Duelle mit dem Gegner beim Tischtennis bzw. Luft-Hockey liefern zu können. Die beiden anderen Sportarten sind aber nicht minder interessant.

Danach wird eine Disziplin ausgewählt und gespielt. Bevor es jedoch um Punkte geht, erscheint ein weiteres Menü und erlaubt beispielsweise die Einstellung von Geschwindigkeit, Schwierigkeitsgrad oder Spieleranzahl.

Das Tischtennis, das besonders aufwendig programmiert wurde und eigentlich schon ein eigenes Spiel wert wäre, bietet noch weitere Einstellungsmöglichkeiten. Viele verschiedene Schlagtechniken können zum Einsatz gebracht werden; z.B.: leichte bzw. feste Schläge gezielt einsetzen und plazieren, um dadurch seinen Gegner von einer Tischkante zur anderen zu hetzen. Gewinnen ist nicht leicht, Übung gehört dazu.

Die grafische Gestaltung, könnte in einigen Details etwas besser sein, kann aber insgesamt als gelungen bezeichnet werden. Zum Sound läßt sich nicht viel sagen, hier wurde gegeizt. Das ist zwar nicht von großer Bedeutung bei Sportspielen, Abstriche müssen deshalb aber gemacht werden.

Fazit

INDOOR SPORTS gehört mit Sicherheit zu den besseren Spielen für den AMIGA. Daß auch zwei Spieler gegeneinander antreten können, und nicht nur der Computer (nicht zu unterschätzen) als Gegner ausgewählt werden kann, wertet das Spiel ungemein auf. Alle Sportarten besitzen einen besonderen Reiz und unterscheiden sich grundlegend voneinander. Dadurch ist für viel Abwechslung gesorgt. INDOOR SPORTS kann allen AMIGA-Besitzern empfohlen werden. Für Sportfans ein Muß in der Software-Sammlung.

Hersteller: Mindscape
Northbrook/USA
Anbieter: GIT GmbH
Oberhöchstädterstr. 53b
6370 Oberursel
Tel. 06171-53863
Preis: DM 90.-



TOP 12

1. SHANGHAI

Das Spielprinzip von SHANGHAI geht auf ein 3000 Jahre altes chinesisches Spiel zurück und verlangt vom Spieler viel Kon-



zentration und Kombinationsgabe, um aus der Steinepyramide gleiche Paare herauszufinden. Am spannendsten sind jedoch die Wettkampfvarianten, bei denen man zu zweit, aber gegeneinander, um Punkte kämpft.

Da ist sie nun, die erste 'TOP 12'-Hitparade der Kickstart-Leser, und sie birgt einige Überraschungen, denn auf den ersten Plätzen tummeln sich nicht die neuen Spiele, sondern bewährte Klassiker wie SHANGHAI, FLUGSIMULATOR II und THE BARD'S TALE. Die Hitparade zeigt aber auch, daß die 'Ballerspiele', die den Markt zur Zeit überfluten, nicht so hoch in der Gunst der Spieler stehen, was auf eine Sättigung schließen läßt. Für die folgende Runde der 'TOP 12' rechnen wir wieder mit Ihrer Teilnahme. Wir verlosen dann auch wieder unter allen Einsendern 12 mal das Spiel, das auf den ersten Platz gewählt wurde. Der Rechtsweg ist dabei ausgeschlossen. Schreiben Sie auf eine Karte Ihr momentanes Lieblingsspiel (wobei jeder Titel gewählt werden kann, den es für den AMIGA gibt!) und schicken Sie die Karte an:

MERLIN Computer GmbH
Redaktion KICKSTART
Kennwort: TOP 12
Postfach 55 69, 6236 Eschborn

oder verwenden Sie die hübsche Karte in der Heftmitte. Einsendeschluß ist der 1. März 1988. Karten, die nach diesem Termin eintreffen, werden im nächsten Monat berücksichtigt.

2. Feary Tale

3. Flugsimulator II

4. The Guild of Thieves

5. The Bard's Tale

6. Indoor Sports

7. Garrison

8. Barbarian

9. Phantasie III

10. Terrorpods

11. Defender of the Crown

12. Plutos

Gewinner je eines SHANGHAI-Spiels sind:
Eike Dehde, Buxtehude / Jürgen Mülbert, Edingen / Frank Unverhau, Bochum / Karola Banowski, Pöring / Ralf Hense, Friedberg / Marc Harekost, Wallenhorst / Lutz Henne-Wellner, Walsdorf / Thomas Sauer, Bonn / Oliver Zellner, Echterdingen / Andreas Grewe, Neumünster / Michael Frese, Düsseldorf / Ulrich Hafner, München

Einkaufsführer

Hier finden Sie Ihren Commodore/Amiga Fachhändler

ANZEIGENSCHLUSS für Heft 5/88 ist der 18.03.1988

3500 Kassel

Hermann Fischer GmbH
Commodore-Systemfachhändler
Rudolf-Schwander-Str. 5-13
3500 Kassel
Tel. (05 61) 70 00 00



4504 GM.-Hütte

dacor Computershop
Auf geht's
Im Sinus-Elektrofachmarkt
Niedersachsenstr. 9
4504 GM.-Hütte
Tel. 0 54 01 / 4 54 41

4650 Gelsenkirchen-Horst



Hard- und Software, Literatur
Bauteile, Service, Versand
Groß- und Einzelhandel
Poststr. 15 · 4650 Gelsenkirchen-Horst
Tel. 0209/52 57 72

4770 Soest

dacor Computershop
Auf geht's
In der Famila-Passage
Senator-Schwartz-Ring 24
4770 Soest · Tel. 0 29 21 / 6 38 03

4800 Bielefeld

hardware
software
organisation
service



CSF COMPUTER & SOFTWARE GMBH
Heeper Straße 106 – 108
4800 Bielefeld 1
Tel. (05 21) 6 16 63

5000 Köln

BÜRO MASCHINEN
braun
AM RUDOLFPLATZ GmbH
5000 KÖLN 1
RICHARD-WAGNER-STR. 39
RUF: 02 21 / 21 91 71

5060 Bergisch-Gladbach

Computer Center
Buchholzstraße 1
5060 Bergisch-Gladbach
Telefon 0 22 02 - 3 50 53

5200 Siegburg

Computer Center
Luisenstraße 26
5200 Siegburg
Telefon 0 22 41 / 6 68 54

5768 Sundern

C.S.C.
Computer & Software-Center
Hauptstr. 2 · 5768 Sundern
Telefon 0 29 33 - 20 46

6000 Frankfurt



Hardware
Software
Service
Computer bco
Büro-Computer + Organisations GmbH
Commodore
TOSHIBA
ATARI OKIDATA
Ihr Partner, wenn es um Computer geht
6000 FRANKFURT/M. 1, Oeder Weg 7-9, ☎ 069/55 04 56/57

6050 Offenbach

CAS-COMPUTER
Ihr Amiga-Spezialist
Sprendlinger Landstr. 71
6050 Offenbach
Tel. 0 69 - 84 20 13

6200 Wiesbaden

Poststraße 25
6200 Wiesbaden-Bierstadt
(061 21) 56 00 84
fax (061 21) 56 36 43



Werbung und EDV GmbH
AUTORISIERTER
COMMODORE
SYSTEM-HÄNDLER



Commodore

6380 Bad-Homburg

PDC GmbH
Produkte u. Details Computerverbund
Louisenstr. 115
Ladenpassage Alter Bahnhof
6380 Bad-Homburg
Tel. 0 61 72 - 2 47 48
Bad-Homburgs erster Commodore Computerladen

6457 Maintal

Landolt-Computer
Beratung · Service · Verkauf · Leasing
Autorisierter Commodore-Händler
Wingertstr. 112 · 6457 Maintal/Dörnigheim
Telefon 0 61 81 - 4 52 93

6551 Fürfeld

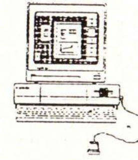


MICHAEL WEISGERBER
HARD + SOFT
RATHAUSSTR. 2
6551 FÜRFELD
TEL: 06709-778

6680 Neunkirchen

Shop 64
Computer GmbH
Saarbrücken * Saarlouis
Homburg * St. Ingbert
Neunkirchen
06821/23713
Commodore Systemhändler

6700 Ludwigshafen



COMPUTING & SOUND
Dieter Hieske
Schillerstrasse 36
6700 Ludwigshafen
Tel. 06 21 / 67 31 05
C — H — S
AMIGA USER ONLY

6800 Mannheim

GAUCH + STURM
Computersysteme + Textsysteme
6800 Mannheim 24
Casterfeldstraße 74-76
☎ (06 21) 85 00 40 · Teletex 6 211 912

6900 Heidelberg

dacor Computershop
Auf geht's
Im Famila-Center
Hertzstraße 1 · 6900 Heidelberg
Tel. 0 62 21 / 30 24 37

6940 Weinheim

dacor Computershop
Auf geht's
Im Mult-Zentrum
Berliner Platz 1 · 6940 Weinheim
Tel. 0 62 01 / 60 01 89

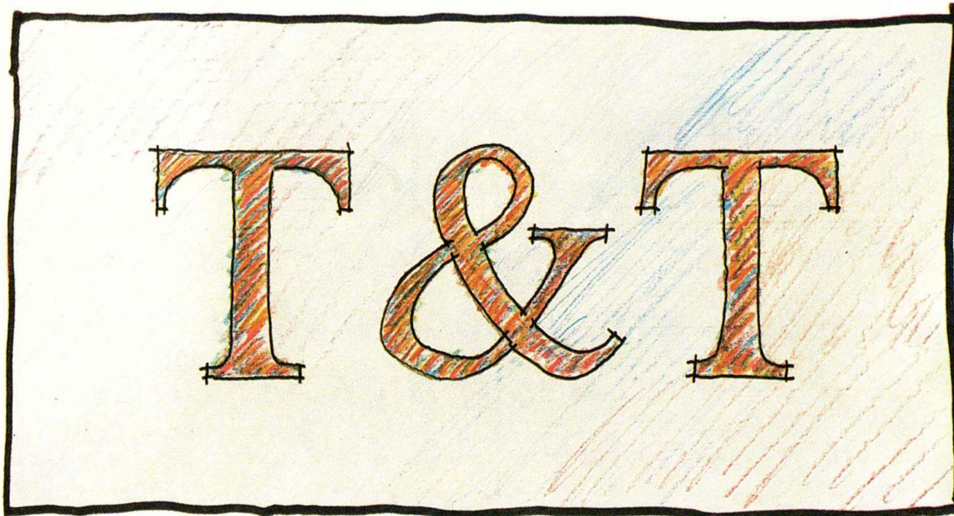
7000 Stuttgart

»If AMIGA, go to Schreiber«
Stuttgart's starker Computer-Laden
SCHREIBER COMPUTER
Alte Poststraße 2
7000 Stuttgart-1
Tel. 0711/22 70 99
Neu · Neu · Neu · Neu
Im SUBWAY
BREUNINGER City

INSERENTENVERZEICHNIS

KICK
START

ArtCom System 1



CLI MIT NEUEM AUSSEHEN

Sicherlich hat den ein oder anderen schon gestört, daß bei manchen Programmen ein vertikaler Streifen auf dem Bildschirm erscheint. Dies geschieht z.B. bei der PD-Diashow von Bionda. Schuld daran ist, daß der Mauszeiger nicht während des Vertical-Blanks ausgeschaltet wird.
Um dies zu verhindern, muß die Sprite-DMA während dieser Zeit umgeschaltet werden. Die nun folgende kurze Assemblerroutine 'NOSPRITE' schaltet die Sprites im richtigen Moment aus:

VERTICAL-BLANK

```
wait:  BTST     #$05,$dff01f     ;Vertical-Blank ?
       BEQ.s    wait             ;nein, weiter warten
       MOVE.w   #$20,$dff096     ;Sprite-DMA aus
       CLR.l    D0
       RTS
```

Damit ist sind alle Sprites und vorallem auch die Maus ausgeschaltet. Um sie wieder zurückzuholen, drückt man entweder <AMIGA>+<N> bzw. <AMIGA>+<M> oder verwendet folgende Routine 'SPRITE':

```
wait:  BTST     #$05,$dff01f      ;Vertical-Blank ?
       BEQ.s    wait              ;nein, weiter warten
       MOVE.w   #$8020,$dff096    ;Sprite-DMA einschalten
       CLR.l    D0
       RTS
```

(Christian Seiler, Unterschleißheim)

Mich ärgerte es , daß ein neu geöffnetes CLI-Fenster immer in der gleichen Form auf dem Bildschirm erscheint und dabei sowohl Disketten-Ikons verdeckt als auch zur Ausgabe von Disketteninhaltsverzeichnissen ungeeignet ist. Mit Hilfe eines Filemonitors (z.B. Filezap von KICKSTART PD 7) kann die Größe und Lage des CLI-Fensters frei definiert werden. Die Änderung muß beim FILEZAP im 8. Record erfolgen. Dort steht der ASCII-Text 'Workbench CON:xx/xx/xxx/xxx/New CLI ...'. Schreibt man dort z.B. die Werte 'CON:00/12/545/244/Kickstart CLI' hinein, meldet sich das CLI beim nächsten Anklicken mit neuem Namen und in einer, so finde ich, wesentlich brauchbareren Form. Zu beachten ist allerdings, daß die gesamte CON-Anweisung nicht länger als 32 Zeichen sein darf, da sich sonst das DOS mit einem 'Unknown Command' meldet.

(Candid Böschen, Petersburg)

NOTEPAD OHNE FONTS

Wer öfters mit dem NotePad gearbeitet hat, dem ist sicherlich auch die lange Ladezeit auf die Nerven gegangen, die bei Verwendung von vielen Fonts auftritt. Deshalb kann es interessant sein, NotePad-Texte ohne Fonts einzulesen. Dazu sind folgende Schritte notwendig:

1. NotePad-Icon anklicken und 'INFO' in der Menüleiste anwählen
2. ADD - Gadget anklicken und in das String - Gadget von 'TOOL TYPES' den folgenden Text eingeben: 'FLAGS = nofonts' und <Return> drücken!
3. INFO mit 'SAVE' verlassen.

Auf eine ähnliche Weise kann auch der voreingestellte Font (Topas 8) bzw. der eigentliche Font des Textes schon beim Laden des NotePads geändert werden. Dazu wird in das String-Gadget von 'TOOL TYPES' der gewünschte Font mit 'FONT =<Fontname>.<Größe>' eingeben.

(Markus Reinhold, Ettenheim)

MONITOR AM AMIGA 2000

Ein Farbmonitor ist nicht gerade billig, und andere Anschaffungswünsche, wie z.B. eine Speichererweiterung oder ein zweites Laufwerk, stehen oft auch noch auf der Wunschliste. Wer aber einen Farbfernseher mit SCART-Buchse besitzt, dessen Bildschirm nicht gerade zu groß ist, ist hier fein heraus. Das normale RGB-Analog-Signal des AMIGA läßt sich nämlich über diese Buchse einspeisen. Die nötige Schaltspannung liefert der AMIGA 2000 am Videoport ebenfalls. Von der Bildqualität her braucht sich der Fernseher nicht zu verstecken. Selbst die 80-Zeichen-Darstellung ist auf meinem Grundig P40-125 gut zu lesen. Beim Interlace-Modus flimmert das Bild sogar weniger als auf manchen Farbmonitoren, da diese oft eine extrem kurze Nachleuchtdauer haben.

Wer jedoch viel im Interlace-Modus oder mit Textverarbeitungen arbeitet, der kann, wenn sein AMIGA 2000 ab der zweiten Jahreshälfte '87 produziert wurde, einen Monochrommonitor direkt anschließen. Neben der Audiobuchse befindet sich nämlich dann eine Buchse mit der Aufschrift 'MONO VIDEO', die ein BAS-Signal liefert.

Wenn man genügend Geld übrig hat, sollte man hier einen Monochrommonitor mit extrem langer Nachleuchtdauer anschließen. Ein solcher Monitor ist zwar für schnelle Spiele nicht geeignet, da die Figuren Streifen hinter sich herziehen, aber für Textverarbeitung und CAD im Interlace-Modus erhält man ein absolut fimmerfreies Bild.

(Thomas Stümpfig, Möglingen)

LEVEL-ENHANCER FÜR EMERALD MINE

Emerald Mine ist zweifelsohne eines der herausragenden Spiele für den AMIGA, nicht nur wegen der hohen Anzahl verschiedener Levels. Um so ärgerlicher ist es für viele Spieler, wenn sie in einem Level 'hängenbleiben' und den Genuß der höheren nicht genießen können da man das nächsten Level erst anwählen kann, wenn man das vorherige geschafft hat.

Mit dem an dieser Stelle abgedruckten Programm wird Abhilfe geschaffen. Es erhöht das Level aller 10 möglichen Single-Player und aller 10 Team-Player. Wenn Sie das Programm abgetippt und gestartet haben, werden Sie aufgefordert, die Emerald Mine-Diskette in das Laufwerk (df0:) einzulegen. Hierzu müssen Sie unbedingt die Original-Diskette verwenden, da eine Kopie von Emerald Mine nur bis Level 4 spielbar ist. Der ungewöhnliche Kopierschutz ist daran schuld. Level 4 ist auf der Kopie nicht zu schaffen, da der Spielfigur unweigerlich ein Stein auf den Kopf fällt. Natürlich ist es nicht besonders ratsam, auf einem Orginal 'herumzupfuschen'. Sicherheitshalber können Sie von der Orginal-Diskette die Datei 'nam' auf eine formatierte Diskette kopieren, um im Falle eines Schreibfehlers die Datei noch in 'Reserve' zu haben. (Es wird nur auf diese Datei zugegriffen !) Legen Sie nach der Aufforderung die Emerald Mine-Diskette ins Laufwerk und drücken Sie eine beliebige Taste. In wenigen Sekunden ist die Modifikation beendet. Jetzt können Sie mit allen 10 Team- und Single-Playern alle Levels erreichen.

Zum Schluß wünsche ich Ihnen viel Spaß mit den weiteren Levels von Emerald Mine.

```
REM *** EMERALD MINE - LEVEL ENHANCER ***
REM *** Autor: Lothar Ziegler          ***
REM *** (C) KICKSTART MAGAZIN 1988     ***
  5:  SCREEN 1,640,200,2,2:
WINDOW 1,"Level - ENHANCER V 1.0",,0,1
LOCATE 10,8
PRINT "Bitte Emerald Mine Diskette in df0: einlegen"
LOCATE 11,8
PRINT "Weiter mit einer beliebigen Taste"
WHILE INKEY$="":WEND
CHDIR "df0:"
OPEN "nam" for INPUT AS 1
  WHILE NOT EOF(1)
    a$=a$+INPUT$(1,1)
  WEND
CLOSE 1
BYTE=14
FOR i=1 TO 20
  READ WERT
  MID$(a$,BYTE)=CHR$(WERT)
  BYTE=BYTE+26
NEXT i
OPEN "nam" FOR OUTPUT AS 1
  PRINT #1,a$;
CLOSE 1
KILL "nam.info"
DATA 182,0,90,148,238,56,114,204,6,80
DATA 170,228,62,136,194,28,86,160,250,52
LOCATE 12,8
PRINT "Bitte warten bis LED erlischt !!!"
LOCATE 15,8
PRINT "Jetzt können Sie Emerald Mine in "
LOCATE 16,8
PRINT "allen Levels spielen."
WHILE INKEY$="" :WEND    37:  SYSTEM
```

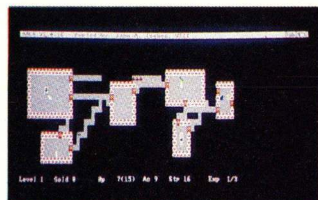
Liebe Leser,
Für den AMIGA gibt es schon eine Unmenge von Public Domain-Programmen, manche Anbieter haben über 100 Disketten in Ihrem Programm. Die verschiedenen Sammlungen sind jedoch zum Teil nicht sortiert und in sich sehr unübersichtlich. Aus diesem Dilemma soll Ihnen der Public Domain Service der KICKSTART helfen. Die einzelnen Disketten werden nach festen Kriterien zusammengestellt, d.h. daß jede Diskette einen Schwerpunkt hat (z.B. Lehrgänge (Tutor), Bilder-Show, C-Programme, Utilities, Spiele, u. ä.). Außerdem werden Angaben über die Programmiersprache, den verwendeten Intepreter oder Compiler usw. gemacht. Die Programme laufen auf allen AMIGA-Computern mit Kickstart/Workbench 1.2, allerdings sollten 512k Speicher vorhanden sein. Sollten dennoch Einschränkungen gelten, so wird dies bei den betreffenden Programmen angegeben.

DAS AKTUELLE ANGEBOT

Diskette 1: C-Source
Eine Sammlung von Programmen, die besonders dem Anfänger zeigen, wie man Intuition programmiert. Die Programme liegen sowohl aus C-Quellkode, als auch als fertige Programme vor, die sofort gestartet werden können.

Diskette 2: Spiele
- **YachtC** (Würfelspiel für 4Personen)
- ***Puzzle***
- **Missile** (verteidigen Sie Ihre Stadt, starker Sound)
- **TriClops** (sehr schönes 3D-Spiel)
- Breakout (3D-Effekt mit Brille)
- **Trek73** (bekannte Star Trek Variante)

Diskette 3: Spiele



HACK: das bekannte Textadventure, das ursprünglich auf UNIX-Rechnern erstellt wurde, liegt hier als spezielle Grafikversion für den Amiga vor.

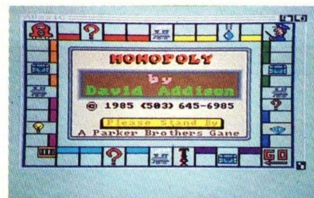
Diskette 4: Terminal-Programme
KERMIT: bekanntes, luxuriöses Terminalprogramm (drei verschiedene Versionen, mit Soucre-Code in C)

Diskette 5: Terminal-Programme
- **WOMBAT** (VT102/52 Emulator, XModem, autodial)
- **VT100** (grafikfähig, Soure in C)
- **TermPlus** (XModem, Soure in C)
- **DG210** (Data General D-210 Emulator)
- **Ahost** (XModem, Kermit)
- **TEK4010** (XModem, VT100)

Diskette 6: Terminal-Programme
- **Speech Term** (spricht den empfangenen Text, XModem)
- **StarTerm** (mit Phone, Duplex, XModem)
- **Argo Term**
- **PD Term** (Source in C)
- **AmigaDisplay**
- **Kermit**

Diskette 7: UTILITIES
- **QuickCopy** (gutes Kopierprogramm)
- **DirUtil** (File-Copy)
- **FileZap** (File-Monitor)
- **DiskZap** (Disk-Monitor)
- **DiskSalv** (Diskettenretter)
- **System-Monitor** - CSH (UNIX-ähnlicher Shell)

Diskette 8: Spiele



Monopoly: sehr schöne Grafik, einfache Mausbedienung (Source in ABasic)

Diskette 9: Grafik-Show
- Grafik-Show mit bekannten Cartoons

Diskette 10: Grafik-Show
JUGGLER DEMO: ein bewegliches Männchen jongliert mit drei verspiegelten Kugeln, sehr schöne Demo

Diskette 11: Grafik-Show
RAY TRACERS: wunderschöne räumliche Bilder, die auf einer VAX berechnet und auf den AMIGA übertragen wurden

Diskette 12: Grafik
- digitalisierte Bilder mit erstaunlicher Qualität (IFF-Format)

Diskette 13: Grafik
- sehr schöne Bilder-Show (IFF-Format)

Diskette 14: EDITOR
bekannter Texteditor MICROEMACS Version 30 viele Features: Search/Replace/ Copy

Diskette 15: Grafik-Animation
Verschiedene Filme, die mit dem AEGIS-ANIMATOR erstellt wurden, incl. PLAYER zum Abspielen der Filme.

Diskette 16: Sprachen
XLISP 1.7 (neueste Version) mit ausführlicher Anleitung (über 50k)

Diskette 17: Sprachen
MODULA-2: Pre-Release eines Modula-Compilers mit verschiedenen kleineren Beispielprogrammen, die als Source-Code vorliegen

Diskette 18: Grafik
MANDELBROT-Generator

Diskette 19: Grafik-Show
sehr schöne, digitalisierte H.A.M.-Bilder

Diskette 20: Grafik-Show
'Fred the Baker und Rose's Flower Shop'
COMIC-Film, der die Multitasking Fähigkeiten des AMIGA erklärt

Diskette 21: AMIGA-Tutor
Einführung in die Bedienung des AMIGA 500. Ein farbenfroher Lehrgang, der ganz von vorne beginnt und mit vielen Bildern und Grafiken die Grundbegriffe des AMIGA erklärt. (für Anfänger, komplett in deutsch)

Diskette 22: Sprachen
MVP-FORTH und C-FORTH
C-Forth ist ein recht leitungsfähiger FORTH-Interpreter, der auch als Quelltext vorliegt.

Diskette 23: Grafik-Show
Viele abwechslungsreiche Motive in verschiedenen Auflösungen, verpackt in einer Grafik-Show.

Diskette 24: Grafik-Show
Sehr schöne, digitalisierte Frauengesichter.

Diskette 25: UTILITIES
CLOCK, PORTAR, MACView, Kickbench, Disassembler, Tracker, Checkmodem, POPCLI und vieles mehr

Diskette 26 & 27: Grafik-Show
Auf zwei randvollen Disketten erleben Sie eine einmalige Dia-Show mit hervorragend digitalisierten, futuristischen Bildern in voller PAL-Auflösung. Dazu gibt es stimmungsvolle, sphärische Musik.

Diskette 28: Editoren
Auf dieser Diskette befinden sich einige schöne Editoren (UEDIT, MED, BLITZ) mit dazugehörigen Zeichensatz-Utilities.

Diskette 29: UTILITIES
PrtDrvGen: erstellt Drucker-Treiber
DropShadow: jedes Fenster bekommt einen Schatten
MemClear: löscht den Speicher
ScreenSave: speichert den Bildschirm auf Diskette
Compress: komprimiert Programme

Diskette 30: SOUND-DEMOS
Digitalisierte Songs: Changing Minds, Joan Lui, Miami Vice II, Respactable, Holiday

Diskette 31: SOUND-DEMO
Dieses Programm erzeugt naturgetreue Geräusche, die über die Tastatur, wie auf einem Klavier, angespielt werden können.

Diskette 32: SOUND-DEMOS
Mit einer Demo-Version von SoundScape können digitaliesierte Musikstücke abgespielt werden. Die Qualität ist wirklich erstaunlich!

Diskette 33: GRAFIK-SHOW



Einige sehr gute, mit Deluxe Video erstellte Filme. Der benötigte PLAYER ist auch auf der Diskette.
INFO: bei AMIGA 500/2000 mit 1MB Speicher erst 'NoFastMem' starten!

Diskette 34: SPIELE
TUNNEL VISION: werden Sie den Weg durch das Labyrinth finden?
REVERSI: eine spielstarke Version des bekannten Brettspiels
KLONDIKE: ein Patience-Kartenspiel

Diskette 35: UTILITIES
ASDG (resetfeste RAM-Disk)
FixDisk, ErrorCk (zur Fehlersuche auf der Diskette)
DiskCat (erstellt eine Übersicht über die Programme Ihrer Disketten)

Diskette 36: CAD
mCAD ist ein wirklich gut gemachtes CAD-Programm, das jedoch nur im Interlace-Modus läuft. Es bietet die einfachen Zeichenfunktionen und Features wie Zoom, Group, Ungroup, Grid, Move, Rotate. Auf der Diskette sind mehrere Dokumente, die das Programm erklären.

Diskette 37: UTILITIES
AddMem: zum Konfigurieren von Speichererweiterungen
MemView: zeigt den Speicherinhalt als Grafik an
GetRom: schreibt das Betriebssystem-ROM des AMIGA 500/2000 als bootfähige Kickstart für den AMIGA 1000 auf Diskette.
MegaPatch: paßt die Kickstart des AMIGA 1000 für das autom. Erkennen von internen Speichererweiterungen an.

Diskette 38: GRAFIK
NoFFP Mandelbrot Set Explorer

DOMAIN SERVICE

V2.1 (neue Version)
von ABC Softarts in Braunschweig

Diskette 39: GRAFIK-SHOW
neue Bilder

Diskette 40: GRAFIK-DEMOS
Boing!, Rotate, Sparks, Moire, Dazzle, 3DCube, Scales, Sizzlers. Sehenswert ist der Film 'Atari meets AMIGA', der die erste und einzige Begegnung der beiden Computer dokumentiert.
Sehr schön ist das Programm LANDSCAPE, das wunderschöne, fraktale Berg- und Tallandschaften erzeugt.

Diskette 41: UTILITIES (Grafik)
Alles, was Sie zu dem von ELECTRONIC ARTS entwickelten Grafik-Standart (IFF-Format) wissen müssen: Laden, Speichern, Komprimieren, Dekomprimieren. Mit Dokumentationen und Source-Codes in C.

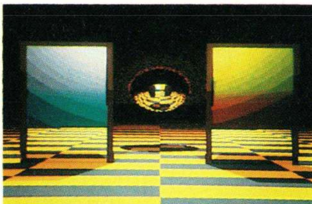
Diskette 42: GRAFIK-SHOW
Vielfältige, nach dem RAY-TRACER-Verfahren erstellte Bilder.
Lassen Sie sich von den realistischen Spiegelungen beeindrucken!
Mit digitalisierter Musik!

Diskette 43: GRAFIK-SHOW
Eine einmalige Show, bei der eine digitalisierte Katze in gleitenden Bewegungen über den Bildschirm trabt. Erstellt wurde diese faszinierende Animation mit einem Digitizer, DPaint und VideoScape 3D.

Diskette 44: SPIELE
Adventurefans kennen sicherlich das Grafikadventure HACK (siehe PD 3). Hier gibt es nun die Fortsetzung: **LARN**. In unüberschaubaren unterirdischen Gängen müssen Gold und Schätze gesucht werden. Aber auch an einem Krafttrunk oder einem magischen Spruch sollte man nicht achtlos vorbeigehen, denn die benötigt man im Kampf gegen Gnome, Vampire und andere Gestalten.
Wirklich sehr empfehlenswert!

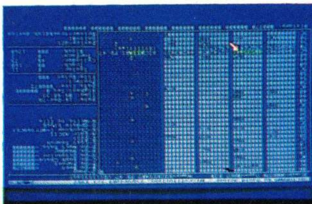
Diskette 45: SPIELE
- **GRAVITYWARS** ist ein interessantes Weltraumspiel, bei dem sich zwei Raumschiffe im Kampf gegenüberstehen. (viele Optionen!).
- **OTHELLO**, eine sehr schöne Reversi-Variante (mit eigenem Fenster!).
- **STREITPATIENCE**, eine Patience-Variante von Hellmut Voelcker (Berlin)
- **CHESS**: spielstarkes Schachprogramm
- **ADVENTURE**: ein Textadventure

Diskette 46: GRAFIK-SHOW



Eine weitere Diskette (siehe auch PD 42) mit phantastischen RAY-TRACER-Bildern, unterlegt mit digitalisierter, fetziger Musik.

Diskette 47: UTILITIES



SECTORAMA: ein sehr nützlicher Disketten- und Festplatten-Monitor, mit dem verlorene oder zerstörte Daten wiederhergestellt werden können.
SILICON: ein sehr komfortabler CLI-Ersatz mit separatem Ausgabe- und Eingabefenster.
DBUG: maschinenunabhängiger Debugger von Fred Fish (Source in C)
TIMER: eine Stoppuhr für die Workbench

Diskette 48: CRAZY
Auf dieser Diskette befinden sich nur verrückte Programme, deren Sinn absolut zweifelhaft ist. Allerdings sollten Sie sich diesen Spaß nicht entgehen lassen!

Diskette 49: ICONS
Utility-Programme, die sich mit der Erstellung und Manipulation von Icons beschäftigen.

Diskette 50: BASIC
Eine Diskette voll Programmen in AmigaBASIC zum Reinschauen, ÇVerändern, Lernen.

Diskette 51: C-Compiler
Ein einfacher C-Compiler, der als Source-Code vorliegt.
Eignet sich für Interessenten der Compilerprogrammierung.

Diskette 52: UTILITIES
Wichtige Helfer, die man haben sollte, denn: "was mer hat, des hat mer, hat mers net, dann fehlt's ahm" (Alt-Frankfurter Sprich wort)!

Diskette 53: COMPILER
Auf dieser Diskette befindet sich die Sprache ADL (Adventure Definition Language). Das System besteht aus Compiler, Interpreter und Debugger, wobei alle Teile als Source in C und auch ablauffertig vorliegen.

Diskette 54: Anwenderprogramme
MICROSPELL (überprüft die Rechtschreibung), ACCESS (Terminal), QBASE (Dateiverwaltung) uvm.

Diskette 55: Grafik/Utilities
Viele schöne Grafikdemos und Utilities zu diesem Thema.

Diskette 56: ASSEMBLER
ASM68K (Macro Assembler mit guter Dokumentation)
ASM (68010 Macro Assembler wie im AmigaDOS Manual beschrieben)
BLINK (bekannter Linker)
AS6502 (portabler 6502-Assembler mit Source in C)

Diskette 57: UTILITIES
Wieder eine der bewährten Überraschungs-Utility-Disketten mit hilfreichen Programmen, die wir speziell dafür aus dem großen Angebot auswählen.

VERSAND BEDINGUNGEN:

Um einen schnellen und problemlosen Versand zu gewährleisten, beachten Sie bitte folgende Punkte.
Lieferung innerhalb einer Woche.

1. Schriftliche Bestellung
- Für jede Diskette ergibt sich ein Unkostenbeitrag von DM 10.-
- Pro Sendung kommt ein Versandkostenbetrag (für Porto und Verpackung) von DM 5.- (Ausland DM 10.-) hinzu.
- Bei Nachnahme zuzüglich DM 3.70 (Nachnahme nur im Inland)
- Bitte legen Sie, falls zur Hand, einen Aufkleber mit Ihrer Adresse bei

2. Anruf genügt

'MERLIN'-Computer GmbH
KICKSTART-Redaktion
Tel.: 06196/481811

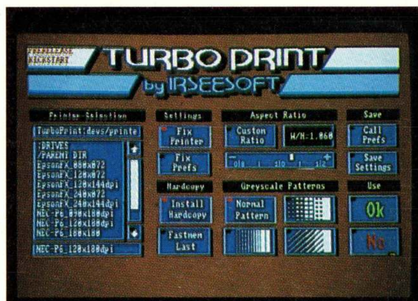
(MO.-FR von 9-17 Uhr)

ANSCHRIFT:

Merlin Computer GmbH
KICKSTART
PD SERVICE
Postfach 5569
6230Eschborn

VORSCHAU 4/88

Turbo Print:
Es ist allgemein bekannt, daß die Druckertreiber der Workbench nicht gerade durch Geschwindigkeit glänzen. Turbo Print will hier Abhilfe schaffen. Der Test wird zeigen, was dieses Programm leistet!



FESTPLATTEN-GRUNDLAGEN:
Neben vielen anderen Grundlagenartikeln, die dem Einsteiger zeigen, wie er mit dem AMIGA umgehen muß, verdient diesmal der Artikel über die Handhabung von Festplatten besondere Beachtung. Es wird aufgezeigt, wie man eine günstige Festplatte im A2000 oder ins Sidecar einbaut.

KICKS für Insider:
Mit dem COLORHUNTER werden 4096 Farben gleichzeitig auf dem Bildschirm dargestellt, die sich mit der Maus auswählen lassen.

MUSIK:
Für die Musik-Fans stellen wir diesmal Soft- und Hardwareprodukte vor. Synthia, ein Synthesizerprogramm, Dynamic Studio, ein Midiprogramm, und das MIDI GOLD-Interface werden getestet.

Und sonst ...
gibt es noch weitere Tests, KICKS, Tips & Tricks und Grundlagen.

Änderungen vorbehalten! Ab 18. März an Ihrem Kiosk!

Impressum

KICKSTART

Chefredakteur:

Uwe Bärtels (Chefredakteur)(UB)
Markus Nerding (Stellvertreter)(MN)

Redaktion:

Andreas Krämer (AK)
Gerald W. Carda (GC)
Harald Schneider (HS)
Marcelo Merino (MM)
Harald Egel (HE)

Herausgeber:

'MERLIN'-Computer GmbH
Industriestraße 26
Postfach 5569
6236 Eschborn
Tel.: 06196/481811
FAX: 06196/41137

Redaktionelle Mitarbeiter:

Christian Keller (CHK)
Andreas Diebold (AD)
Wolf Dietrich (WD)
Andreas Suchy (AS)
Jobst Hermeier (JH)

Autoren dieser Ausgabe:

Florian du Bois
Michael Brinkmann
Sebastian Dosch
Roland Foerster
Ralf Görlach
Mathias Kegelmann
Ernst Heinz
Herbert Kunz
Robert Malzan
Edgar Meyzis
Roger Schmidt
Oliver Siebenhaar
Martin Silbernagl
Rainer Spirandelli
Sven Stillich
Georg Tiefenbacher
Thomas Trost
Lothar Ziegler

Public Relations:

Claus Peter Lippert

Auslandskorrespondenz:

D. dela Fuente (GB)
L. Hennely (USA)

Verlag:

Heim Verlag
Heidelberger Landstraße 194
6100 Darmstadt 13
Tel.:06151/56057
FAX:06151/55689
FAX:06151/56059

Verlagsleitung:

Hans-Jörg Heim

Anzeigenverkauf:

Kyriakulla Margaritis
Uwe Heim (Ltg.)

Anzeigenpreise:

nach Preisliste Nr.3, gültig ab 1.1.88

Layout, Titelbild, DTP

Fabian & Mayer

Produktion:

Karl-Heinz Hoffmann
Klaus Schultheis
Susanne Failer
Bernd Failer

Fotographie:

Rainer Spirandelli, Archiv

Titelfoto:

Rainer Spirandelli

Druck:

Ferling Druck, Darmstadt

Bezugsmöglichkeit:

Zeitschriftenhandel, Kauf- und Warenhäuser, Commodore-Fachhändler oder direkt beim Verlag.

KICKSTART erscheint 11 mal im Jahr
Einzelpreis:
DM 7,- ,ÖS 56,- SFr 7,-
Jahresabonnement Inland:
DM 70,-
Europ. Ausland DM 90,-
Luftpost DM 120.-



Amigo Laufwerke

AMIGO FB1	(Bausatz einer 3.5" Einzelfloppy)	275.-
AMIGO FB1	(s. o. jedoch mit Gehäuse)	297.-
AMIGO F1	(anschlußf. 3.5" Einzelfloppy)	329.-
AMIGO F2	(anschlußf. 3.5" Doppelfloppy)	599.-
AMIGO FB2	(Bausatz einer 3.5" Doppelfloppy)	539.-
AMIGO FB2	(s. o. jedoch mit Gehäuse)	559.-
AMIGO 2000	(internes Zweitlaufwerk Amiga 2000)	259.-
AMIGO /NEC	(Rohlaufwerk 1036a 3.5")	247.-
AMIGO **Gehäuse**	für eine 3.5" Floppy	24.90
AMIGO **Gehäuse**	für zwei 3.5" Floppies	35.90
AMIGO **Gehäuse**	für eine 5,25" Floppy	24.90
AMIGO **Bootselector**	DF1:/DF2:	19.90

AMIGA-HARDWARE '88

RAM Erweiterung 2 MB für Amiga 2000 (interne Steckkarte incl. Systemtest und resetsichere Ramdisk)	847.-
RAM Erweiterung 2 MB für Amiga 1000	947.-
RAM Erweiterung 2 MB für Amiga 500 (mit Netzgerät)	997.-
RAM Erweiterung 0.5 MB für Amiga 500 (abschaltbar mit Uhr, in abgeschirmten Gehäuse)	277.-
Filecard 20MB (Tandon)	687.-
Monitor Commodore 1081/1084	687.-
Monitor Eizo Flexscan 8060S	1549.-
Monitor Nec Multisync	1387.-
Monitor Mitsubishi 1471a	1398.-
Digi View V2.0 PAL	275.-
TV Modulator	67.-
Digi View Gender Changer	49.-
Perfect Sound (Stereo Sounddigitizer)	167.-
Amiga 2000 RAM Karte 2MB	948.-
Amiga 2000 Hard Disk (Supra) 20MB	1398.-
Amiga 2000 AT Karte	1898.-
Amiga 2000 DMA Harddisk Interface	449.-
Amiga 500 SCSI Controller	549.-
Amiga 500 20MB Harddisk incl. Controller	1898.-
Amiga 500 30MB Harddisk incl. Controller	2098.-

!!! Unser Angebot !!!

3.5" Disketten 2DD ab 100 Stück je	2.27
Aegis Vidoscape 3D	277.-
Digi View V2.0 PAL	275.-
Digi Paket PAL (Digi View 2.0 & Digi Paint)	347.-

Bitte beachten: Wir nehmen kurzfristige Preissenkungen vor!!!
Weitere Artikel in unserem Gesamtkatalog. Neuerscheinungen aus unserer Sonderliste.

Da bei einem durchgeschleiften Floppybus die effektive Kabellänge zum Drittlaufwerk mehr als die zulässigen 70cm beträgt, werden unsere Laufwerke auf Wunsch mit einem **durchgeschleiften Floppystecker** ausgerüstet und nicht am Gehäuse durchgeschleift!

Soyka DATEN TECHNIK

ANWENDUNGS-SOFTWARE

AC Basic	387.-	Introcad	137.-
AC Fortran	597.-	Keybord Cadett	77.-
Acquisition	597.-	**K Seka Assembler**	137.-
Aegis Animator mit Images	257.-	LPD Writer	198.-
Aegis Art Pak	57.-	**Marauder II**	67.-
Aegis Draw Plus	437.-	MCC Pascal	227.-
Aegis Images	77.-	Metacomco Shell	147.-
Aegis Impact	147.-	Metacomco Toolkit	97.-
Aegis Sonix	147.-	Modula II Commercial	597.-
Aegis Videotitler	167.-	Modula II Developers	297.-
Aesop's Fables	97.-	Modula II Regular	177.-
AMIGA Assembler	187.-	Music Studio	97.-
AMIGA C	257.-	Newio	497.-
Animate 3D	257.-	Page Setter	327.-
Aztec C Commercial	888.-	Pixmate	117.-
Aztec C Developers	487.-	Printmaster Plus	97.-
Cambridge LISP	397.-	Prowrite	247.-
City Desk	277.-	Professional Page	697.-
CLI Mate V1.2	67.-	Project D	87.-
Deluxe Music Construction	197.-	Publisher 1000	347.-
Deluxe Paint II	197.-	Scribble Plus	187.-
Deluxe Print	187.-	Sculp 3D	187.-
Deluxe Video V1.2	197.-	Superbase	227.-
Devpac Assembler	197.-	True Basic	297.-
Digi Paint	117.-	TV Text	187.-
Dynamic CAD	888.-	UBM Text V2.2	147.-
Dynamic Studio	347.-	**Videoscape 3D**	277.-
Express Paint	157.-	**Vizawrite**	197.-
Flipside	97.-	**Word Perfect V4.1**	687.-
Grabbit	67.-	Write & File	147.-
Instant Music	87.-	Zing	147.-

UNTERHALTUNGS-SOFTWARE

Alien Fires	77.-	Impact	47.-
Amiga Karate	67.-	Jagd auf roter Oktober	67.-
Arazok's Tomb	77.-	Kampfgruppe	107.-
Archon II	77.-	**Karate Kid II**	67.-
Backlash	67.-	Karate King	37.-
Bad Cad	57.-	Karting Grand Prix	27.-
Balance of Power	77.-	**King of Chicago**	87.-
Barbarian	67.-	Knight Orc	77.-
Bard's Tale	87.-	Leader Board	67.-
Beat it!	27.-	Leisure Suit Larry	87.-
Borrowed Time	67.-	Leviathan	67.-
Breach	67.-	Marble Madness	67.-
Breach Scenerydisk	37.-	Mean 18	87.-
Bridge 4.0	57.-	Mindbreaker	27.-
Bureaucrazy	87.-	Mission Elevator	57.-
Challenger	27.-	Moebius	77.-
Championship Baseball	67.-	Ninja Mission	37.-
Championship Basketball 2	67.-	Pac Boy	27.-
Championship Football	67.-	Phalanx	27.-
Championship Golf	67.-	Quintett	87.-
Chessmaster 2000	77.-	Quiwi	57.-
City Defence	27.-	Rockett Attack	27.-
Crystal Hammer	37.-	**Shadowgate**	87.-
Cruncher Factory	27.-	Shanghai	77.-
Dark Castle	57.-	Shooting Star	27.-
Defender of the Crown	77.-	Sindbad	77.-
Demolition	27.-	Sky Fighter	57.-
Donald Duck's Playground	57.-	Space Battle	27.-
Dr. Fruit	27.-	Space Fight	27.-
Earl Weaver Baseball	97.-	Star Glider	77.-
Emerald Mine	27.-	Stellar conflict	77.-
Faery Tale	77.-	Strange new world	47.-
Feud	37.-	Strip Poker (Artworx)	77.-
Flightsimulator II	97.-	**Terrorpods**	67.-
Flip Flop	27.-	Test Drive	67.-
Football Fortunes	67.-	The Big Deal	77.-
Fortress Underground	27.-	The Final Trip	27.-
Garrison	67.-	The Pawm	87.-
Gee Bee Air Rally	67.-	The Surgeon	137.-
Goldrunner	77.-	Trivia	27.-
Grand Slam	87.-	Typhoon	67.-
Gridstart	27.-	Ultima	77.-
Guild of Thieves	87.-	Ultima II	77.-
Fire Power	57.-	Ultima III	77.-
Hacker II	77.-	Uninvited	77.-
Hardball	97.-	Wintergames	67.-
Hellowoon	67.-	World Games	67.-
Hollywood Poker	47.-	Xenon (Neu)	57.-

Demnächst in unserem Angebot

Deluxe Print II	177.-
Deluxe Write	197.-
Music X	537.-
Publisher Plus	287.-
Autoduell	87.-
California Games	77.-
Clever & Smart	67.-
Diana Sisters	57.-
Ferrari Formula One	77.-
Garrison II	67.-
Go	77.-
Gunship	87.-
International Karate	67.-
Into the Eagles Nest	67.-
Land of Legends	87.-
Planetarium	97.-
Return to Atlantis	87.-
Star Wars	67.-
Street Gang	57.-
Turbo	47.-
Wizball	67.-

Soyka DATEN TECHNIK
Wir führen die komplette Amiga Soft- und Hardware
Hattinger Str. 685 · 4630 Bochum 5
Telefon 0234/41 19 13
41 19 47

Entwurf + Druck PLUM, Kerpen-Sindorf, Tel. (02273) 53091

GOLEM

KUPKE

Wir liefern im 3-Tage-Rhythmus



02 31/81 83 25-27
Telefax 02 31/81 74 29
D-4600 Dortmund 1
Burgweg 52a

1 Golem Drive 3,5

NEC 1036a mit heller Frontblende ● Amiga-farbenes Metallgehäuse ● Abschalter ● Busdurchführung bis DF3 ● PC-Karten und Sidecar kompatibel **!!! neu !!! jedes Drive mit Trackdisplay** zur aktuellen Spur- und Kopfanzeige

mit Display DM **379.-**
ohne Display DM **369.-**

2 Golem Drive 5.25

5,25 Zoll Laufwerk mit heller Frontblende ● Amiga-farbenes Metallgehäuse ● Abschalter ● Busdurchführung bis DF3 ● 40/80 Track Umschalter Amiga und MS-Dos kompatibel **!!! neu !!! Drive mit Trackdisplay wie Golem 3,5**

mit Display DM **449.-**
ohne Display DM **439.-**

3 Golem Drive 3,5 intern

modifiziertes NEC 1036a mit heller Blende ● Staubschutzklappe zum Einbau in den A 2000 incl. Einbausatz und Einbauanleitung

DM **269.-**

4 Golem Ram Box 1000

2 MB Erweiterung ansteckbar ● in Amiga-farbenem Metallgehäuse ● Abschalter ● Busdurchführung ● auto konfigurierend ● Betriebskontrollanzeige durch LED ● erweitert den Grundspeicher auf 2,5 Megabyte

DM **998.-**

Technische Änderungen vorbehalten

5 Golem 500

Ram Erbeiterung speziell für den Amiga 500 ● technische Einzelheiten wie Golem Ram Box 1000 ● beide Erweiterungen ohne Wait States

DM **998.-**

6 Kickstart / Uhr Modul

"Bitte Workbench einlegen", so meldet sich ihr Amiga 1000 mit dem Kickstart Eprom Modul ● Ansteckbar am Systembus ● Amiga-farbenes Metallgehäuse ● durchgeführter Systembus ● abschaltbar sodaß andere Kickstart Versionen wieder gebootet werden können.

DM **199.-**

Amiga 2000 u. 500 kompatibeles, externes Uhrenmodul ansteckbar am Systembus ● Software, die die 2000/500 Uhr anspricht, benutzt auch die Golem Clock für den A 1000

DM **149.-**

Uhr und Kickstart in einem Gehäuse DM **299.-**

7 Golem Sound

Audio Digitizer der Spitzenklasse, kompatibel zu aller gängigen Software mit DIN- und Cinch-Anschluß auch für Micro Anschluß geeignet ● optische Aussteuerung über ein LED Display ● STEREO ● Wandlungsfähig ● 1MHz getaktet bietet der Golem Sound unglaubliche Sample Qualität.

Mono DM **139.-**
Stereo DM **189.-**
Software zum Golem Sound, stereofähig DM **29.-**

Entwurf Firma DataCom, Dortmund · Realisierung, Plum, 5014 Kerpen-Sindorf, Tel. 022 73 5 30 91